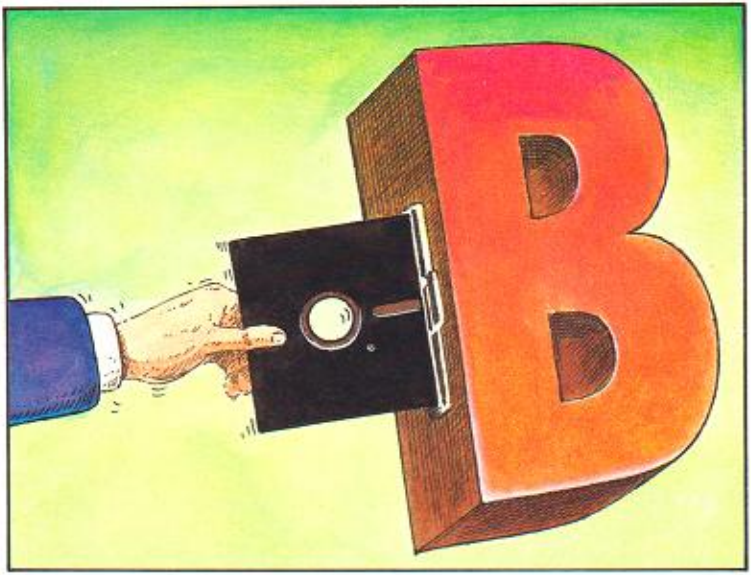
TO 7 MO5 TO7 70

LE BASIC D-O-S DU TO7/TO7-70 ET DU MO5

Christine et François-Marie Blondel



TO TEK International
cedic nathan

LE BASIC D.O.S. DU TO7, TO7-70 ET DU MO5

Christine et François-Marie Blondel
Illustrations : Dominique Izoard

TO TEK International
cedic
nathan

Dans la même collection

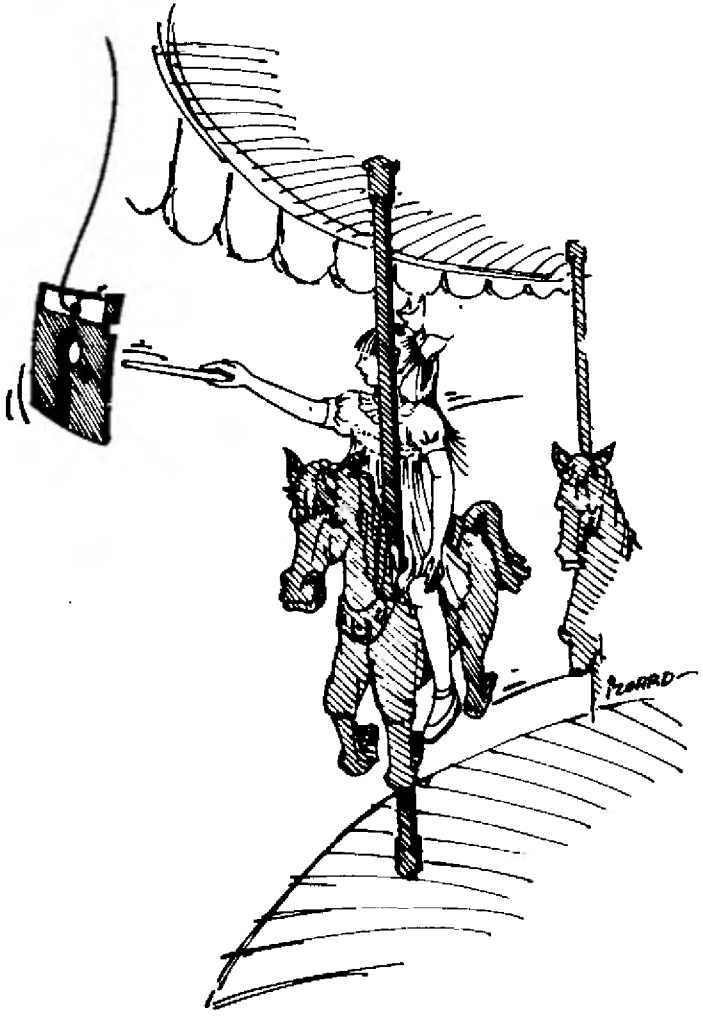
Initiation au BASIC TO 7 — *Christine et François-Marie Blondel.*
Un ordinateur en fête — *Serge Pouts-Lajus.*
Un ordinateur et des jeux — *Jean-Pascal Duclos.*
Un ordinateur à la maison - *Jean Delcourt*
Manuel Technique du TO 7 et TO 7-70 — *Michel Oury*
Guide du MO 5 — *André Deledicq*
Initiation à LOGO — *Doris Avram - Michèle Weidenfeld*
LOGO, Manuel de référence — *Doris Avram - Tristan Savatier - Michèle Weidenfeld*
Manuel de l'Assembleur 6809 sur TO7/TO7-70 — *Michel Weissgerber*
Initiation au FORTH sur TO7/TO7-70 et MO5 — *S.E.F.I.*

Ce volume porte la référence
ISBN 2-7124-0520-X
ISBN 2.7124-0584-6 -
2e édition revue et corrigée

Toute reproduction, même partielle, de cet ouvrage est interdite. Une copie ou reproduction par quelque procédé que ce soit, photographie, photocopie, microfilm, bande magnétique, disque ou autre, constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1957 sur la protection des droits d'auteur.

CEDIC 1983

CEDIC, 32, boulevard Saint-Germain. 75005 - PARIS



Avant-propos

Avec un (ou plusieurs) lecteurs de disquettes et le D.O.S. BASIC, vous venez d'augmenter considérablement les capacités de votre micro-ordinateur.
Le D.O.S. BASIC, ce sont deux disquettes identiques qui contiennent des compléments au BASIC de la cartouche, en particulier en graphique, et toute la partie du BASIC qui permet de gérer des fichiers sur disquette (D.O.S signifie Disk Operating System).
L'objectif de cet ouvrage est de servir à la fois de manuel d'initiation et de manuel de référence. Vous y trouverez expliquées à partir d'exemples concrets toutes les nouvelles instructions graphiques dans un chapitre particulier. Tout le reste de l'ouvrage est consacré à l'utilisation des disquettes : mettre en route, enregistrer des programmes puis stocker des données dans les deux types de fichiers qui sont offerts, fichiers à accès séquentiel et fichiers à accès direct.
Le souci de traiter toutes les instructions et fonctions relatives aux fichiers nous a conduit à aborder des détails qui ne sont pas indispensables lors d'une première utilisation. Certaines parties vous apparaîtront comme moins importantes, au moins dans un premier temps. Vous pouvez les survoler, elles se trouvent en général en fin de chapitre. En particulier, la fin du chapitre sur les fichiers à accès séquentiel n'est pas du tout indispensable pour savoir utiliser de tels fichiers.
Bien que l'ouvrage soit rédigé de façon (presque) continue, vous n'êtes pas tenu de tout lire à la suite. Après le premier chapitre, vous pouvez choisir de lire ce qui vous intéresse le plus : c'est un manuel à entrées-sorties multiples !
La composition des exemples suppose une connaissance minimale de BASIC ; reportez-vous au manuel d'initiation ou au manuel de référence si besoin est. Cependant, pour ceux qui connaissent déjà bien les fichiers, nous avons placé en annexe une description complète de toutes les commandes, instructions et fonctions sur fichiers.
La totalité du livre est destinée tant aux utilisateurs de TO 7 ou T0 7-70 qu'aux utilisateurs de MO 5. Les renseignements qui ne concernent qu'une seule machine sont signalés dans le texte, le plus souvent par le nom du micro-ordinateur entre parenthèses : (TO 7) ou (MO 5).
Notre souhait est que chacun trouve ici les réponses à ses questions, et le plus facilement possible.
Maintenant, à vos disquettes !

Sommaire

Chapitre I — Mettre en route le système 9
— pas de ratés à l'allumage 9
— disquette système et disquettes vierges 10
— comment est faite une disquette ? 10
— précautions vis-à-vis des disquettes 12
— chargement du Basic Disque 13
— extinction du système 15

Chapitre II — Enregistrer ses programmes sur disquette 17
— l'initialisation d'une disquette (DSKINI) 17
— recopier le contenu d'une disquette sur une autre disquette (BACKUP) 20
— enregistrer un programme sur disquette et le charger en mémoire centrale (SAVE, LOAD) 22
— modifier un programme enregistré sur disquette 23
— fusionner deux programmes (MERGE) 24
— la consultation du catalogue (DIR) 25
— changer le nom d'un programme (NAME) 26
— mesurer la place libre sur une disquette (DSKF) 27
— éliminer un programme de la disquette (KILL) 27
— recopier un seul fichier (COPY) 28
— une sage précaution (VERIFY) 29
— utiliser plusieurs lecteurs de disquettes 29
— le périphérique pris par défaut (DEVICE) 31

Chapitre III — Les nouvelles instructions du Basic Disque ... 33
— de cercle en cercle : une cible (CIRCLE) 33
— des cercles à volonté 35
— un demi-cercle pour un ballon 36
— remplir le ballon... de rouge (PAINT) 38
— des cercles pleins 39
— dessiner à la boussole (DRAW) 41
— un moulinet en couleurs 43
— petit intermède sur l'affichage des couleurs à l'écran 44
— lever le crayon en cours de dessin 44
— changer d'échelle 46
— tracer en aller-retour 48
— tourner le dessin d'un quart de tour 49
— une étoile qui scintille encore plus 50
— des DRAW les uns à l'intérieur des autres 52

— mémoriser et déplacer une image (GET, PUT) 53
— autres compléments (HEX$, SPACE$, STRING$, DEF FN, DEF USR, MID$, SWAP, AUTO, RENUM, SEARCH).... 54

Chapitre IV — Les fichiers séquentiels 61
— qu'est-ce qu'un fichier de données ? 61
— la gestion d'un fichier séquentiel 62
— écrire un enregistrement 63
— lire l'enregistrement qu'on vient d'écrire 64
— écrire et lire plusieurs enregistrements 66
— un nombre quelconque d'enregistrements 68
— modifier un enregistrement 69
— un fichier de nombres : les points d'un dessin 72
— utiliser plusieurs fichiers de données de même nature 73
— de l'importance de la fermeture des fichiers (UNLOAD) .. 74
— recensement général des périphériques utilisables avec un fichier 75
— voir le fichier s'écrire à l'écran 77
— recopier un fichier sur cassette 78
— et si votre lecteur de disquettes est en panne 79
— gagnez de la place en mémoire grâce à PRINT *9* 80
— utilisez PRINT *9* USING pour bien présenter vos données sur imprimante 83
— règle de bon usage 85

Chapitre V — Les fichiers à accès direct 87
— une chose pour chaque jour, un jour pour chaque chose : un agenda simplifié 87
— plusieurs données dans un enregistrement 93
— modifier la taille d'un enregistrement 93
— découper un enregistrement en plusieurs morceaux 93
— une facilité d'écriture dans PUT *9* et GET *9* 95
— relire les chaînes de caractères 96
— effacer le contenu d'un enregistrement 97
— écrire des données numériques 97
— si les nombres sont entiers 99
— Utilisation de LSET et RSET sur MO 5 100

Annexes :
1. Description des instructions et fonctions de gestion des fichiers sur disque et des nouvelles instructions et fonctions du Basic Disque 103
2. Organisation d'une disquette 125
3. Les fonctions USR en langage machine 129
4. Liste des codes d'erreurs 133

Index 141

Chapitre I / Mettre en route le système

Le lecteur enregistreur de disquettes (on dit aussi "unité" de disquettes), a plusieurs fonctions :
— enregistrer des programmes, comme le lecteur de cassettes, mais de manière infiniment plus rapide et plus souple ;
— stocker et gérer des données (listes ou tableaux de nombres ou de mots) en plus grande quantité que dans la mémoire de l'unité centrale : la capacité de stockage d'une disquette est de 80 K octets. Ceci, encore une fois, de manière beaucoup plus souple qu'avec un lecteur de cassettes ;
— enfin grâce au Basic Disque plusieurs instructions et fonctions nouvelles s'ajoutent au Basic de la cartouche.
Dans ce premier chapitre, nous allons voir comment mettre en route le "système", c'est-à-dire l'ensemble unité centrale et ses périphériques (téléviseur, lecteur de disquettes, éventuellement imprimante, manettes de jeu...).
Ensuite nous "chargerons" le Basic Disque, c'est-à-dire le contenu de la disquette système, dans la mémoire centrale. Vous aurez alors deux possibilités :
— aller au chapitre II pour voir comment enregistrer vos programmes sur disquette ;
— aller au chapitre III pour découvrir les nouvelles instructions et fonctions du Basic Disque.

Pas de ratés à l'allumage

Si vous avez réussi à trouver une prise multiple, accessoire indispensable à tout utilisateur de micro-ordinateur (et à surveiller jalousement), vous allez pouvoir brancher tous les fils qui sortent de votre système informatique.
— TO 7 : n'oubliez pas l'extension mémoire !
— TO 7 et TO 7-70 : mettez en place la cartouche BASIC, et raccordez le lecteur de disquettes à l'un des trois connecteurs situés sur la partie arrière gauche de l'unité centrale.
— MO 5 : raccordez le lecteur de disquettes à l'unique connecteur situé derrière l'appareil.
D'une façon générale, tout périphérique, toute extension doivent être en place avant que l'unité centrale soit allumée.

Ordre d'allumage

1. le téléviseur,
2. le lecteur de disquette *vide* (à priori il vous est vendu sans disquette à l'intérieur, surtout n'en mettez pas). Les voyants vert et rouge s'allument et un bruit de moteur sort du lecteur. Le voyant vert s'éteint au bout de quelques secondes (durée de rotation du moteur) tandis que le voyant rouge reste allumé tant que le lecteur est sous tension,
3. les autres périphériques éventuels,
4. l'unité centrale.

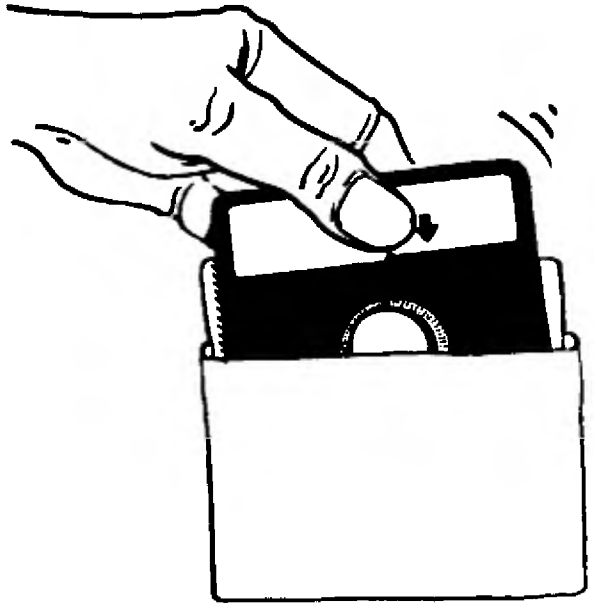
— Vous possédez un TO 7 ou un TO 7-70 : le "menu" s'affiche à l'écran.
— Vous possédez un MO 5 : BASIC affiche son message de bienvenue à l'écran.
Un peu de patience... Lisez les trois paragraphes suivants avant de continuer la mise en route de la machine.

Disquette système et disquettes vierges

Vous avez sans doute aussi à votre disposition un paquet de disquettes vierges (disquettes à 1 trou d'index ou à "sectorisation logicielle", voir plus loin).
Sans la disquette système (BASIC DOS), le lecteur ne peut rien faire : cette disquette contient les instructions et les commandes qui permettent d'utiliser les autres disquettes.

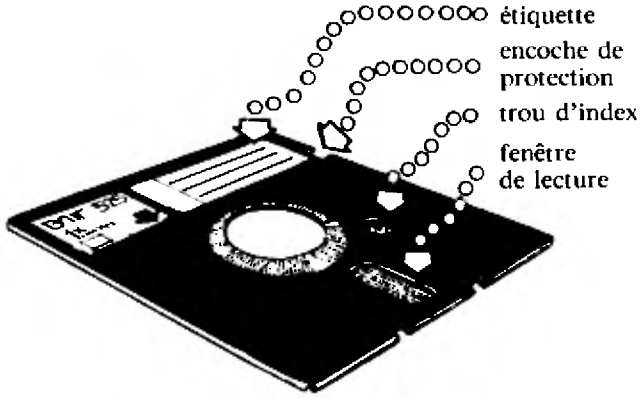
Comment est faite une disquette ?

D'abord elle est placée dans une pochette en papier à conserver. Il n'y a que deux endroits possibles pour une disquette : à l'intérieur du lecteur ou dans sa pochette en papier.
Si vous la sortez (voir dessin), vous constatez qu'elle est entièrement protégée dans une enveloppe de carton.
Cette enveloppe étant inamovible (et ne tournant pas !), la disquette tourne à l'intérieur de l'enveloppe dans le lecteur.
La disquette est ainsi protégée car elle est très souple (en anglais "floppy disk" = disque souple), beaucoup plus souple qu'un disque de musique. C'est une mince galette de mylar recouverte d'une couche uniforme d'oxyde magnétique, comme une bande magnétique, mais sur toute la surface.
En effet, la "lecture" sur la disquette se fait avec une tête de lecture identique à celle d'un lecteur de cassettes. Ce procédé de lecteur magnétique est donc tout à fait différent de la lecture sur un disque microsillon. Sur ce dernier est gravé un sillon en spirale légèrement irrégulier et dont les variations de forme font vibrer une aiguille : le procédé de lecture est alors mécanique.



Une seule face de la disquette magnétique est utilisée : celle du côté de l'étiquette. Toutes les disquettes sont toujours introduites avec la face utile (celle de l'étiquette) sur le dessus.
Si vous entendez dire "c'est un cinq pouces", c'est que le système décimal n'est pas encore une référence universelle, surtout dans le pays à la source de la technologie informatique : l'enveloppe mesure 5 pouces 1/4 de côté soit environ 13,3 cm. Il existe des disques magnétiques qui suivent d'autres standards (par exemple 8 pouces).

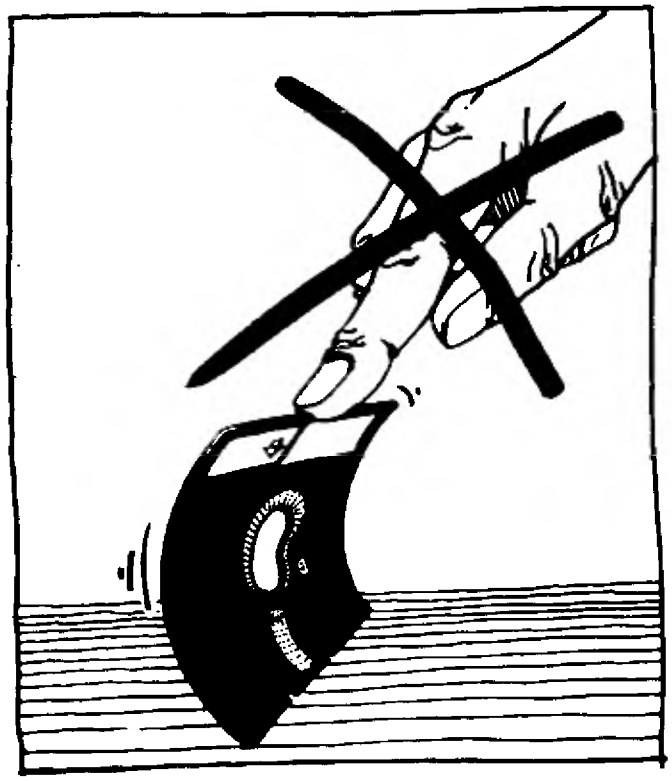
La partie centrale circulaire de la disquette est souvent renforcée : c'est par cette partie centrale que se fait l'entraînement de la disquette à la vitesse de 300 tours/minute (contre 33 tours/minute pour un microsillon). Lors du prochain allumage du lecteur vous pourrez jeter un coup d'œil par l'ouverture et voir le cylindre central en rotation lorsque le moteur tourne.
L'ouverture allongée au travers de laquelle apparaît la surface de la disquette est la fenêtre de lecture. C'est par cette fenêtre que la tête de lecture peut venir au contact de la surface de la disquette. Le petit trou circulaire permet au lecteur de repérer un trou correspondant dans la disquette qui sert d'index pour définir le "début" de la disquette.



Sur les disquettes (sauf l'une des deux disquettes système) il y a encore une découpe dans le carton sur le côté droit : c'est l'"encoche de protection". Il suffit d'y coller une des petites étiquettes auto-collantes fournies avec les disquettes (ou à défaut un morceau de ruban adhésif) pour empêcher l'écriture sur cette disquette. Tant que l'encoche de protection est recouverte on peut seulement "lire" le contenu de la disquette. Une des deux disquettes système est protégée, mettez-la de côté. Vous pouvez également protéger la deuxième, celle que vous utiliserez, afin de ne pas détruire son contenu par inadvertance. Nous l'utiliserons toujours "en lecture".

Précautions vis-à-vis des disquettes

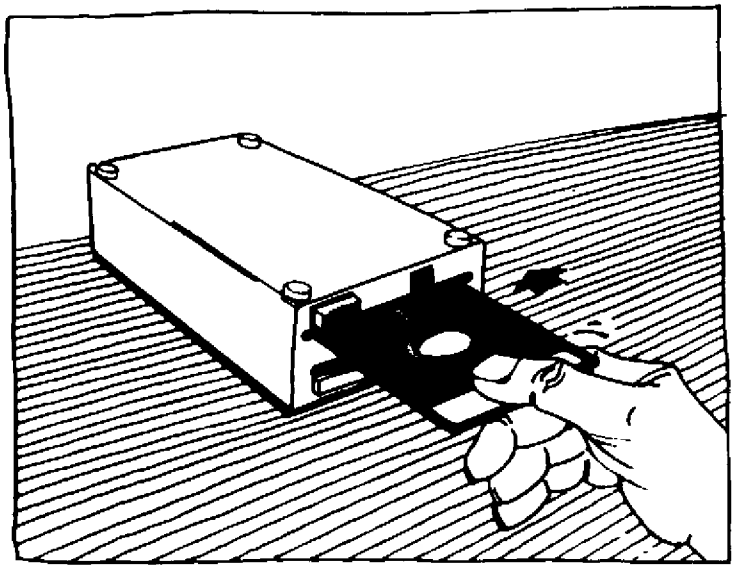
Les disquettes magnétiques sont comme des pêches bien mûres : fragiles et délicates. Et même encore plus sensibles puisqu'il ne faut jamais poser les doigts ou tout autre objet sur la surface magnétique, ne pas les plier, faire attention à la poussière, à la fumée de cigarettes (ne parlons pas des cendres), aux boissons, alcoolisées ou non...
Une disquette sortie du lecteur doit réintégrer sa pochette et, lorsqu'elle n'est plus utilisée, être mise à l'abri.
En outre, la disquette étant magnétique, il faut la tenir éloignée de tout champ magnétique. Si nous ne pouvons pas échapper au champ magnétique de la terre, celui-ci est suffisamment faible pour ne pas faire courir de risques aux disquettes. Par contre un téléphone, un vieux tournevis ou pire évidemment un aimant ont de fortes chances de détruire l'enregistrement fait sur une disquette placée à proximité.



C'est le même phénomène qui se produit lorsqu'un ticket de métro est laissé longtemps au contact de pièces de monnaie ou de clefs. Si, dans ce cas, le ticket de métro est devenu inutilisable, par contre une disquette a des chances de pouvoir servir à nouveau (mais son contenu est perdu).
Toutes ces précautions peuvent vous paraître un peu superflues et ennuyeuses, mais il faut mieux éviter des destructions malencontreuses. Il n'est pas agréable de perdre le fruit de dizaines, parfois de centaines d'heures d'effort !

Chargement du Basic Disque

Revenons au chargement du Basic Disque :
— Placez la disquette système dans le lecteur, étiquette sur le dessus.
— Fermez le loquet qui empêche de sortir la disquette du lecteur. La fermeture du loquet amène la tête de lecture en contact avec la disquette. Ne jamais ouvrir la porte lorsque le voyant vert est allumé.



Maintenant deux cas peuvent se présenter.:
1. Si vous possédez un TO 7 ou un TO 7-70 :
— appuyez sur le chiffre [1] pour répondre au menu. Le lecteur tourne et au bout de quelques instants apparaît à l'écran le message de bienvenue "DISK BASIC".
2. Si vous possédez un MO 5 :
— le BASIC résident est déjà actif, il vous faudra taper :

DOS

Cette commande va charger les informations supplémentaires du BASIC à partir de la disquette système. Attendez le message "OK" pour continuer à travailler.
Le Basic Disque est "chargé" dans la mémoire de l'unité centrale (UC) : le contenu de la disquette système a été transféré en mémoire centrale. La mémoire centrale étant "volatile", c'est-à-dire que son contenu "s'envole" lorsqu'on éteint l'UC, il faut "recharger" le Basic Disque en mémoire centrale à chaque fois que l'UC a été éteinte.
En fait le chargement est terminé dès que le message "OK" apparaît à l'écran, donc avant l'arrêt de la rotation du moteur et l'extinction du voyant vert. Il en sera ainsi dans toutes vos opérations avec le lecteur : le message OK indique que l'opération est terminée.
Vous avez maintenant à votre disposition :
— l'interpréteur BASIC en "mémoire morte" (cartouche)
— une copie de la disquette système en "mémoire vive".
Vous pouvez alors sortir la disquette système du lecteur et la ranger dans son enveloppe.

Vous verrez, ultérieurement, en Annexe 2, qu'il est possible d'exécuter directement un programme sur disquette, au chargement du Basic Disque.

Extinction du système

1 — Retirer la disquette du lecteur.
2 — Éteindre l'unité centrale.
3 — Éteindre le lecteur, le téléviseur...

Règles de sécurité :

- A chaque fois qu'on allume ou éteint l'UC ou un périphérique, le lecteur doit être VIDE,

- A chaque fois qu'on allume ou éteint un périphérique, l'UC doit être ÉTEINTE.

Vous pouvez maintenant poursuivre avec le chapitre suivant (Instructions pour l'enregistrement de programmes sur disquettes) ou le chapitre III (Nouvelles instructions BASIC contenues dans la disquette système).

Chapitre II / Enregistrer ses programmes sur disquette

Dans ce chapitre nous allons voir comment exploiter le lecteur de disquettes pour enregistrer (et relire) une première catégorie d'informations : les programmes. Dans la suite nous verrons comment gérer la deuxième catégorie d'informations : les fichiers de données. Ici le lecteur de disquettes jouera donc le même rôle que le lecteur de cassettes, mais avec beaucoup plus de rapidité, de souplesse... et de sécurité.
Les précautions signalées dans le chapitre I pour la disquette système sont évidemment valables pour toutes les disquettes, en particulier ne jamais éteindre le lecteur ou l'UC avec une disquette dans le lecteur.
Le Basic Disque étant chargé, vous pouvez retirer la disquette système du lecteur.

L'initialisation d'une disquette (DSKINI)

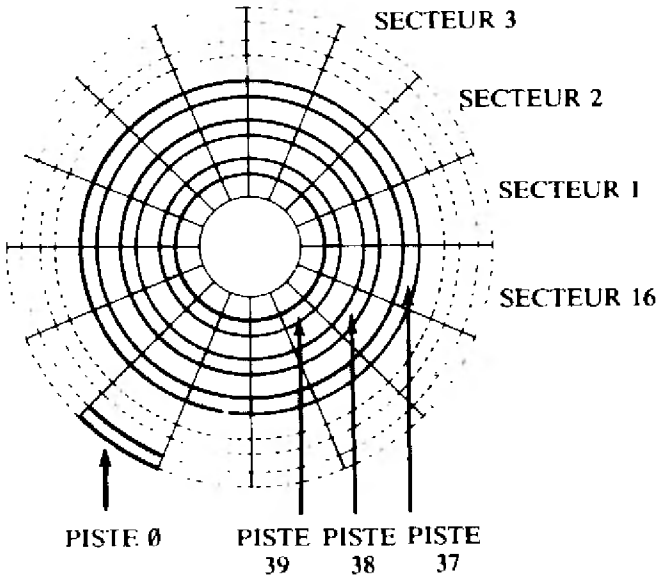
Avant d'enregistrer quoi que ce soit pour la première fois sur une disquette, il faut préparer sa surface : c'est l'initialisation.
ATTENTION ! Cette opération détruit le contenu de la disquette si celle-ci n'est pas vierge. Il faut donc manipuler avec précaution l'instruction d'initialisation. Évitez par exemple de l'employer avec la seconde disquette système, sauf si vous souhaitez réellement effacer tout ce qu'elle contient !
Placez donc une disquette NEUVE, étiquette sur le dessus, dans le lecteur. Si vous avez plusieurs lecteurs, reportez-vous plus loin au paragraphe "Utiliser plusieurs lecteurs de disquettes".
Pour initialiser la disquette, tapez :

DSKINI Ø

Ø est le numéro du premier lecteur de disquettes, obligatoire même s'il n'y a qu'un lecteur.
La syntaxe complète est la suivante (voir Annexe 1) :

DSKINI n° lecteur de disquettes concerné, entrelacement.
L'entrelacement est optionnel (implicitement fixé à 4).

Pendant la durée de l'initialisation (environ une minute), voyons ce qui se passe.
Sur une cassette, le stockage de l'information est très simple : les informations sont inscrites les unes à la suite des autres. L'inconvénient évident est que pour accéder à une information donnée, il faut dérouler toute la bande jusqu'à l'endroit cherché.
Au contraire avec une disquette, la tête de lecture peut se placer très rapidement au-dessus de n'importe quel point de la surface à condition de pouvoir repérer ces différents points... et de savoir à quel endroit se trouve ce que l'on cherche. C'est le rangement des données sur la disquette qui explique la rapidité et la facilité d'accès à l'information.
Le stockage de l'information ne se fait pas de façon continue comme sur une bande magnétique ou sur le sillon en spirale d'un disque ordinaire, mais sur des anneaux ou "pistes" successifs.



Chaque piste correspond à une position bien déterminée de la tête de lecture au-dessus de la fenêtre de lecture de la disquette. La tête de lecture de votre lecteur est réglée pour occuper 40 positions différentes successives : ce lecteur définit donc 40 pistes sur la disquette.
L'initialisation consiste en gros :
1 — A faire passer la tête de lecture (en écriture) sur chaque piste en la divisant en 16 secteurs égaux (chaque secteur correspond à 1/16e de tour). Un secteur contient lui-même 128 octets, c'est la quantité minimale de mémoire échangée entre la disquette et l'unité centrale. La capacité de la disquette est donc :
128 × 16 × 40 = 81920 octets soit 80 K octets (80 K).
La durée de l'initialisation représente le temps nécessaire pour que la tête de lecture initialise les 81920 octets, c'est-à-dire la durée d'enregistrement et de vérification, de toute la capacité mémoire de la disquette.
2 — A organiser un endroit sur la disquette pour son "catalogue", c'est-à-dire la liste du contenu de la disquette.
L'opération est terminée, et votre disquette prête à l'emploi dès que le message OK apparaît à l'écran. Le moteur continue à tourner un certain temps après, comme pour chaque opération de lecture ou d'écriture.

Remarques :
Pour distinguer une disquette initialisée d'une disquette vierge, vous pouvez y coller une étiquette sur laquelle vous écrirez ultérieurement le contenu.
Un moyen plus systématique est d'en demander le catalogue comme nous le verrons plus loin.
Si vous initialisez involontairement deux fois de suite une même disquette neuve, cela ne présente comme seul inconvénient que de vous faire perdre une minute.
L'initialisation est aussi un bon moyen de "nettoyer" une disquette sur laquelle vous avez déjà enregistré des programmes ou des fichiers de données dont vous n'avez plus besoin.

Recopier le contenu d'une disquette sur une autre disquette (BACKUP)

Non seulement les disquettes sont fragiles et délicates, mais encore elles sont aussi, comme nous, sujettes à l'inéluctable phénomène du vieillissement : à la longue, le frottement de la tête de lecture sur la surface de la disquette use le revêtement magnétique.
Si la duplication génétique des individus fait encore partie de la science-fiction, par contre la reproduction des disquettes est très simple à effectuer.
Nous appellerons disquette "*source*" la disquette que nous voulons recopier et disquette "*destination*" celle sur laquelle la recopie sera effectuée. Cette dernière peut être vierge (mais initialisée) ou déjà enregistrée mais dans ce cas bien sûr, son contenu sera perdu.
Un des premiers usages de la copie de disquettes est de faire une duplication de la disquette système. Vous en avez reçu deux exemplaires, ce qui vous garantit contre un éventuel défaut de fabrication dans une disquette.
Cependant il est souhaitable de ne pas utiliser celle qui est protégée en écriture et de ne travailler qu'avec des copies. Ceci vous permettra de revenir à l'original en cas de défaillance.
En outre vous pouvez utiliser des copies de la disquette système pour enregistrer vos programmes et données. En effet le Basic Disque n'occupe que 8 K sur les 80 de la disquette.
Nous allons donc copier la disquette système protégée en écriture (disquette "*source*") sur une disquette vierge préalablement initialisée par DSKINI (disquette "*destination*").
Comment faire avec *un seul* lecteur pour transférer de l'information d'une disquette sur une autre ?
La seule solution est de "charger" la disquette "*source*" dans la mémoire centrale, retirer cette disquette du lecteur, la remplacer par la disquette "*destination*" et enfin enregistrer le contenu de la mémoire centrale sur la disquette.
En fait ceci n'est pas possible en une seule opération puisque la capacité en mémoire de la disquette est très supérieure à celle de la mémoire centrale. La copie se fera donc morceau par morceau.
Précautions indispensables avant de recopier une disquette :
— s'assurer que l'on ne veut pas garder le programme présent en mémoire centrale, car l'on détruit ce qui pouvait s'y trouver au préalable ;
— protéger la disquette "*source*" en écriture. Ainsi si l'on intervertit par mégarde les deux disquettes (cela arrive...), on ne risque pas de recopier la disquette "*destination*" sur la disquette "*source*" effaçant ainsi ce qu'on veut précieusement sauvegarder.

Placez la disquette système protégée en écriture dans le lecteur.
Tapez alors :

BACKUP 0

(*to back up* = soutenir quelqu'un)
0 désigne le numéro du premier lecteur de disquettes, numéro obligatoire même s'il n'y a qu'un lecteur.
S'il y en a plusieurs, reportez-vous plus loin au paragraphe "Utiliser plusieurs lecteurs de disquettes".
BACKUP 0 est une syntaxe réduite utilisée lorsque l'on travaille sur un seul lecteur ou sur le premier s'il y en a plusieurs. La syntaxe complète est (voir Annexe 1) :

BACKUP n° de lecteur de la disquette "*source*" TO n° de lecteur de la disquette "*destination*".

Au-delà de quelques secondes de bruit pendant lesquelles l'ordinateur charge dans l'UC une partie de la disquette, apparaît sur l'écran le message :
Insert destination diskette and press "ENTRÉE".
Retirez alors la disquette "*source*" du lecteur et insérez la disquette "*destination*" puis appuyez sur ENTRÉE. Après un nouveau ronronnement de quelques secondes, pendant lesquelles l'ordinateur transfère sur la disquette le contenu de la mémoire centrale, un nouveau message apparaît à l'écran :
Insert source diskette and press "ENTRÉE"
Échangez à nouveau les deux disquettes.
Il faut une dizaine de ces opérations pour transférer tout le contenu de la disquette "*source*" sur la disquette "*destination*", ceci prend environ trois minutes.
Ne croyez pas réduire la durée de la copie parce que votre disquette "*source*" est presque entièrement vide.
Certes, le cerveau humain a besoin d'un quart de seconde pour voir si une page est blanche et en déduire qu'il faut la tourner, mais l'ordinateur doit lire ligne après ligne, caractère après caractère , même s'ils sont tous blancs !
Le seul moyen de simplifier le processus de copie est de satisfaire aux règles de la consommation en allant acheter un deuxième lecteur de disquettes ! Cela se justifie si vous avez de gros fichiers de données à gérer.
Au bout de trois minutes d'assouplissement des poignets la sauvegarde est faite. Retirez la disquette du lecteur et éteignez le système (en respectant l'ordre d'extinction).

Puis reprenez la mise en route et chargez le Basic Disque à partir de la disquette de sauvegarde. Tout doit se passer comme avec la disquette d'origine.
La place disponible sur cette troisième Disquette Système pour enregistrer vos programmes est bien sûr un peu moins importante qu'avec une disquette vierge.

Enregistrer un programme sur disquette et le charger en mémoire centrale (SAVE, LOAD)

Cette opération ne vous posera pas beaucoup de problèmes puisqu'elle s'effectue de la même façon qu'avec le lecteur de cassettes.
Voici un petit programme :

```
100 ' NOTES
110 PLAY "[illegible]"
```

Pour l'enregistrer, il suffit de taper :

```
SAVE "NOTES"
```

Un nom de programme peut contenir au maximum huit caractères. Dès que OK apparaît à l'écran, le programme est transféré sur la disquette et l'endroit où il est stocké (numéro de piste et numéros de secteurs) est inscrit dans le catalogue.
Si vous obtenez le message d'erreur ND Error (No Disk : pas de disquette), ou Error 71, c'est sans doute que vous avez oublié de fermer le loquet du lecteur. Si par mégarde vous essayez de sauvegarder un programme sur la disquette système protégée en écriture, vous obtiendrez un autre message d'erreur : WP Error (Write protected = disquette protégée en écriture) ou Error 72.
Lorsque le programme est enregistré, vous pouvez vider la mémoire centrale :

```
NEW
RUN
```

puis "charger" le programme :

```
LOAD "NOTES"
RUN
```

ou de manière équivalente :

```
RUN "NOTES"
```


Cette forme de l'instruction RUN commande à la fois le chargement et l'exécution.
Vous pouvez ajouter un suffixe de 3 caractères à la suite du nom du programme (le programme et le suffixe étant séparés par un point).
Par exemple pour un autre programme de musique :

```
100 ' SYMPH5
110 PLAY "[illegible]"
120 PLAY "[illegible]"
```

```
SAVE "SYMPH5.MUS"
```

Si vous ne précisez pas de suffixe, l'ordinateur en ajoutera un automatiquement ; "BAS" pour "programme en Basic".
Pour recharger ce programme, il faudra donc faire :

```
LOAD "SYMPH5.MUS"
```

afin d'éviter à l'ordinateur de rechercher "SYMPH5.BAS" qui n'existe pas.

Modifier un programme enregistré sur disquette

Si vous enregistrez un programme sous un nom de programme déjà utilisé, le premier programme est effacé et remplacé par le deuxième. Attention aux lapsus avec l'instruction SAVE ! Mais ceci est par contre très utile pour modifier un programme enregistré.
Modifions par exemple le programme NOTES :

```
LOAD "NOTES"
[illegible]
```

```
SAVE "NOTES"
```

Vérifions le bon enregistrement du programme modifié :

```
NEW
LOAD "NOTES"
RUN
```

Fusionner deux programmes (MERGE)

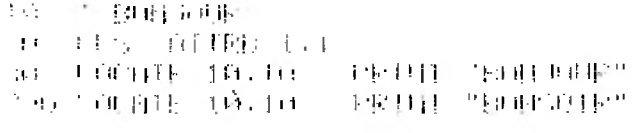
Un même indicatif musical peut être joué dans plusieurs jeux, une page écran présentant un "menu" au crayon optique peut servir dans plusieurs programmes, de même qu'un calcul de TVA, de prix de revient... Ce sont des parties de programmes qu'il faut alors réécrire dans chaque nouveau programme qui les utilise. Comment éviter cette recopie ?
L'instruction MERGE (= fusionner) permet de charger un programme enregistré sur disquette (ou sur cassette) en mémoire centrale sans effacer le programme qui y est (le programme "résident"). Ceci à deux conditions :
1. que le programme enregistré ait été sauvegardé sous forme de texte en ASCII. Inutile de chercher ce qui se passe dans la disquette. Chargez "SYMPH5.MUS" puis tapez :

SAVE "SYMPH5", A

pour que le programme SYMPH5 soit sauvegardé sous forme ASCII.
2. que les deux programmes à fusionner, celui qui est en mémoire centrale et celui de la disquette, n'aient pas de numéros de lignes communs. Si c'est le cas, c'est la ligne du programme enregistré qui l'emporte.
Entrez maintenant en mémoire centrale un nouveau programme (qui ne contienne pas de lignes 100, 110 et 120).

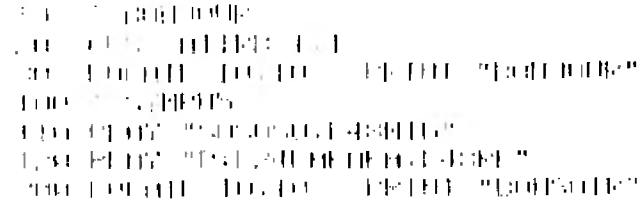
Par exemple :

NEW



Puis chargez SYMPH5 en mémoire centrale, sans effacer BONJOUR, en tapant :

MERGE "SYMPH5"
LIST



Vous pouvez maintenant enregistrer la nouvelle version de BONJOUR sur disquette. Si vous voulez que personne ne découvre comment est rédigé ce programme extraordinaire, sauvegardez-le sous forme "protégée". Sous cette forme on pourra exécuter le programme mais non le lister : la suite des instructions restera secrète.

SAVE "BONJOUR", P

Attention : Si vous n'en avez pas gardé — en lieu sûr — une copie non protégée, vous ne pourrez plus jamais lister ni modifier le fruit de vos longues heures d'effort !

La consultation du catalogue (DIR)

Rien n'est à vendre, tout est à faire dans ce catalogue : c'est à vous de le remplir avant de le feuilleter.
Voyons où en est notre disquette. Tapez :

DIR

La réponse est le "catalogue" ("directory" = catalogue) qui donne la liste de tous les programmes enregistrés sur la disquette.

NOTES	BAS	B	B	1
SYMPH5	MUS	B	B	1
SYMPH5	BAS	B	A	1
BONJOUR	BAS	B	B	1
nom du programme	suffixe	B = programme en basic D = données	B = binaire A = ASCII	volume occupé en blocs

Le 1er programme enregistré "NOTES" n'avait pas de suffixe : le suffixe BAS lui a été attribué "par défaut" (à défaut d'autre chose).
Le 2e programme "SYMPH5" a été enregistré une première fois avec le suffixe MUS et une deuxième fois en ASCII mais sans suffixe.
Dans la 3e colonne, "B" signifie : programme en Basic. Avec un fichier de données, on obtient ici la lettre D.
Dans la 4e colonne, "B" signifie : programme stocké en binaire. Les programmes sont stockés en binaire sauf lorsqu'on utilise l'instruction SAVE..., A, c'est-à-dire qu'on les sauve en ASCII, ce qui est le cas du programme SYMPH5.BAS.

La dernière colonne indique le volume occupé par chaque programme en nombre de "blocs", un bloc étant formé de 8 secteurs (soit une demi-piste ou 1K). En fait ces petits programmes sont loin de remplir un bloc entier mais l'attribution de la place sur le disque se fait par blocs entiers.
La consultation du catalogue n'efface pas le programme en mémoire centrale, vous pourrez l'utiliser à tout moment pour contrôler les enregistrements et modifications effectués.
Elle vous permettra aussi de savoir si une disquette a été initialisée ou non. Avec une disquette vierge DIR donne :
UD Error (Unreadable Diskette : disquette non initialisée) ou Error 76.
Avec une disquette initialisée (mais vide), DIR donne :
OK

Pour arrêter le déroulement momentanément agissez sur STOP, pour l'arrêter complètement frappez CNT C.
La syntaxe complète de DIR est (voir Annexe 1) :

DIR "n° de lecteur de disquettes :". Cette dernière mention n'est à spécifier que s'il y a plusieurs lecteurs.

Changer le nom d'un programme (NAME)

Il est souvent intéressant de conserver au moins temporairement des versions successives d'un même programme, surtout lorsque l'enregistrement est aussi simple et rapide.
Si par exemple le programme BONJOUR ne vous satisfait plus, ou que vous voulez avoir le choix entre plusieurs versions possibles du programme BONJOUR, vous pouvez l'appeler BONJOUR1 :

```
NAME "BONJOUR.BAS" AS "BONJOUR1.BAS"
```

(= changer le nom de "BONJOUR" en "BONJOUR1".)
N'omettez pas d'indiquer le suffixe BAS après les noms, sinon les programmes BASIC ne seront pas reconnus dans l'instruction NAME.

Vérifiez le résultat de l'opération avec :

```
DIR
```

Si vous proposez dans l'instruction NAME comme nouveau nom un nom qui a déjà été attribué à un programme, vous obtenez un message d'erreur qui vous évite d'"écraser" involontairement ce programme :

AE Error (Already Exists : existe déjà) ou Error 66.

Mesurer la place libre sur une disquette (DSKF)

Vous pouvez initialiser une deuxième disquette vierge pour mesurer la place maximum disponible sur une disquette :

```
DSKINI Ø
```

(Ø représente le numéro du lecteur).
S'il y en a plusieurs, reportez-vous plus loin au paragraphe :
"Utiliser plusieurs lecteurs de disquettes".

Et une minute plus tard...

```
PRINT DSKF(Ø)
78
```

DSKF (Ø) est le nombre de blocs libres sur le lecteur n° Ø, ici 78 blocs, soit 78 K. A deux K octets près, on retrouve la capacité mémoire de la disquette (8ØK). Ces deux K octets sont justement réservés pour écrire le catalogue de la disquette.
Pour la disquette sur laquelle vous avez enregistré vos programmes, le calcul est facile à faire : 78 K au départ, 1 K pour chacun des quatre programmes, il doit donc en rester 74. Vérifions-le :

```
? DSKF (Ø)
74
```

Lorsque la disquette comporte une vingtaine de programmes de tailles différentes et que vous voulez savoir s'il reste assez de place pour enregistrer un vingt et unième programme, il sera plus simple d'utiliser la fonction DSKF (n° lecteur) qui est l'équivalent pour la disquette insérée dans ce lecteur de l'instruction FRE(Ø) pour la mémoire centrale.

Éliminer un programme de la disquette (KILL)

Lorsque vous êtes satisfait de votre dernière version, il ne vous reste plus qu'à jeter au panier les ébauches successives, ou du moins une partie d'entre elles. C'est facile si vous avez numéroté ces versions dans le nom des programmes (BONJOUR1, BONJOUR2...) ou si vous avez daté vos programmes. Cette dernière précaution est toujours utile : elle permet de connaître la succession des différents programmes.
Pour effacer de la disquette un programme qui ne sert plus, et récupérer la place correspondante, il suffit de donner l'ordre :

```
KILL (Tue) !
```

Mais avec cette instruction, comme avec NAME, il est indispensable de préciser le suffixe du programme.
Lorsque vous n'avez précisé aucun suffixe lors de la sauvegarde, le programme a reçu automatiquement le suffixe .BAS :

KILL "NOTES.BAS"

Vérifions le résultat de ce crime avec :

DIR

Recopier un seul fichier (COPY)

Alors que l'instruction BACKUP permet la copie intégrale d'une disquette indépendamment de son contenu, l'instruction COPY permet de faire la copie d'un seul programme (ou d'un seul fichier de données).
Pour copier le programme "NOTES", sur une autre disquette, il suffit d'écrire (cas d'un seul lecteur) :

COPY "NOTES.BAS"

Le suffixe (BAS ici) est indispensable dans l'instruction COPY, comme dans KILL et NAME.
Lors de l'exécution de l'instruction, l'ordinateur lit le programme sur la disquette située dans le lecteur, puis vous demande d'introduire la disquette sur laquelle vous désirez effectuer la copie, et lorsque vous avez appuyé sur ENTRÉE, recopie le programme sur la disquette destination.
Vous pouvez aussi recopier un programme sur la même disquette, mais cette fois vous devrez changer son nom. Par exemple :

COPY "BONJOUR.BAS" TO "BONSOIR.BAS"

recopie sur la même disquette le programme "BONJOUR" en le renommant "BONSOIR".
Vous pouvez le vérifier sur le catalogue.

Une sage précaution (VERIFY)

Nous avons déjà insisté (un peu trop peut-être) sur la fragilité de l'enregistrement magnétique sur disquette. De fait les incidents sont rares, mais leurs conséquences fâcheuses. Une disquette qu'on ne peut plus relire : ce sont des heures passées à réécrire les programmes qui s'y trouvaient, s'ils n'avaient pas été conservés par ailleurs.
Mais les défauts d'écriture peuvent être plus subtils : un caractère seulement peut avoir été mal enregistré.
Il existe une commande spéciale de vérification de l'écriture sur disquette.
Si vous tapez :

VERIFY ON

toutes les opérations d'écriture (SAVE, COPY, BACKUP, DSKINI) que vous effectuerez par la suite feront l'objet d'une vérification. Par exemple, lors de l'enregistrement d'un programme, après la phase d'écriture sur la disquette, le programme est relu et comparé à ce qui se trouve en mémoire.
Cette vérification allonge un peu le temps nécessaire à l'exécution de SAVE ou COPY, mais c'est en général négligeable.
La durée de DSKINI est sensiblement doublée ; c'est le prix d'une vérification complète.
Il vous est possible de supprimer ces vérifications à tout moment : il suffit de taper :

VERIFY OFF

(Lorsque vous chargez le Basic Disque, c'est l'option VERIFY OFF qui est prise implicitement).

Utiliser plusieurs lecteurs de disquettes

Si vous avez plusieurs lecteurs de disquettes (votre micro-ordinateur peut en gérer jusqu'à quatre), de nombreuses opérations vont se trouver facilitées. Et surtout, vous disposerez de 2, 3 ou 4 fois plus de données simultanément disponibles en fichier.
Attention : il faut avoir connecté et mis sous tension tous les lecteurs si vous voulez les voir actifs au moment voulu.

Le descripteur de fichier

Jusqu'à présent nous avons désigné les fichiers programmes par leur nom : "SYMPH5.MUS", "BONJOUR"...
Avec plusieurs lecteurs de disquettes, il devient indispensable de préciser, en plus du nom, le numéro du lecteur sur lequel le programme se trouve. Le numéro de lecteur se place devant le nom du fichier et il en est séparé par deux points. Le premier lecteur (le plus bas) est le numéro Ø, le suivant le numéro 1 et le dernier le numéro 3.
Ainsi si vous voulez enregistrer le programme "JEU" sur le deuxième lecteur, il vous faut écrire :

SAVE "1:JEU"

"1:JEU" est le *descripteur* du fichier programme "JEU" et pour le charger :

LOAD "1:JEU"

Un coup d'œil sur le catalogue du lecteur n° 1 s'obtiendra par :

DIR "1:"

Pour connaître la place encore libre sur cette disquette, tapez :
? DSKF (1)

De même pour les instructions NAME et KILL, il faut utiliser le descripteur de fichier :

NAME "2:TORTUE.BAS" AS "2:LIEVRE.BAS"

change le nom du programme "TORTUE" (dans le lecteur n° 2) en "LIEVRE".
La sauvegarde d'une disquette complète (avec BACKUP) devient beaucoup plus simple avec deux lecteurs de disquettes.
Pour copier la disquette contenue dans le lecteur n° Ø sur la disquette contenue dans le lecteur n° 1, il vous suffit de taper :
BACKUP Ø TO 1, ainsi que le montre la syntaxe complète, et d'attendre que cela se passe...
Vous risquez beaucoup moins d'erreurs qu'en intervertissant les disquettes (n'oubliez pas cependant de protéger la disquette "*source*" en écriture — un malheur est si vite arrivé...).

Avec plusieurs lecteurs de disquettes (2 par exemple), la copie d'un seul fichier se fera aussi simplement par :

COPY "Ø:NOTES.BAS" TO "1:NOTES.BAS".

Le périphérique pris par défaut (DEVICE)

Si vous travaillez le plus souvent avec le lecteur n° 1, une instruction vous évitera de répéter à chaque opération le numéro du lecteur à utiliser.
Lorsque vous ne précisez pas de numéro dans le descripteur de fichier, c'est le lecteur n°Ø qui est automatiquement utilisé.
C'est le périphérique "implicite" ou "pris par défaut" :

SAVE "JEU"

Le programme en mémoire est enregistré sur le lecteur n° Ø avec le nom JEU.BAS.
Tapez la commande :

DEVICE "1:"

puis à nouveau :

SAVE "JEU"

C'est le 2e lecteur qui s'allume. Le programme en mémoire s'est enregistré sur le lecteur n° 1.
Faites :

DIR

Vous obtenez le catalogue du lecteur n° 1.
Ainsi l'instruction DEVICE "1:" (= "périphérique") permet de changer de périphérique implicite : dans toutes les commandes qui ne précisent pas le numéro de lecteur, c'est le lecteur n° 1 qui sera choisi.

En *résumé*, nous venons de voir l'essentiel des commandes qui permettent d'utiliser les disquettes pour enregistrer des programmes.
Dans les chapitres suivants, nous verrons comment créer et utiliser des fichiers de données.
Les commandes et fonctions DIR, DSKINI, DSKF, BACKUP et VERIFY concernent toute la disquette indépendamment de son contenu : fichiers programmes ou fichiers de données. Les commandes NAME, KILL, COPY s'utilisent aussi bien pour un fichier programme que pour un fichier de données.
Quant aux instructions LOAD, SAVE, MERGE elles sont réservées aux programmes et nous allons voir dans le chapitre IV ce qui joue le même rôle pour les fichiers de données.

Chapitre III / Les nouvelles instructions du Basic Disque

De cercle en cercle : une cible (CIRCLE)

Une cible est une série de cercles concentriques. Les cibles de foire n'ont que deux couleurs, mais ici nous pouvons utiliser les huit couleurs disponibles du BASIC sur TO 7 (moins une couleur pour le fond) et donc tracer sept cercles de couleurs différentes.
Voici un programme qui permet de dessiner un cercle :

```
10  ' CERCLE
20  CLS
30  COULEUR=3
40  C0=160 : L0=100     ' CENTRE
50  R=80                : ' RAYON
60  PSET(C0+R,L0),COULEUR
70  FOR T=0 TO 6.3 STEP 0.1
80    C=C0+R*COS(T)
90    L=L0+R*SIN(T)
100   LINE -(C,L),COULEUR
110 NEXT T
```

Adaptez-le pour tracer sept cercles concentriques (gardons le centre de l'écran, colonne CØ = 160, ligne LØ = 100) et de rayons 10, 20, 30... 70 avec par exemple les couleurs correspondant aux chiffres des dizaines : 1, 2, 3... 7. La huitième couleur (Ø = noir) sera celle du fond. C'est l'intérêt de placer les différentes quantités qui interviennent au cours d'un programme dans des variables (ici les coordonnées du centre CØ, LØ, le rayon R et la couleur COULEUR) : il est très facile de réutiliser le même programme avec des valeurs différentes.
Ici nous aurons 7 tracés de cercle avec les 7 couleurs successives et le rayon R = 10 * COULEUR.

```
10  ' CIBLE1
20  CLS:SCREEN 7,0,0
30  C0=160 : L0=100
40  FOR COULEUR=1 TO 7
50    R=10*COULEUR : ' RAYON
60    ' Trace d'un cercle
      de centre (C0,L0) et de rayon R
70    PSET(C0+R,L0),COULEUR
```

```
80     FOR T=0 TO 6.3 STEP 0.1
90        C=C0+R*COS(T)
100       L=L0+R*SIN(T)
110     LINE -(C,L),COULEUR
120    NEXT T
130 NEXT COULEUR
```

La première instruction supplémentaire de Basic Disque permet de tracer un cercle sans avoir recours à un programme : c'est l'instruction CIRCLE.
Essayez :

RAZ

CIRCLE (160, 100), 70, 7

les coordonnées du centre sont 160, 100
le rayon est 70
la couleur est 7

Avec cette instruction nous pouvons tracer la même cible : il suffit de remplacer la suite des lignes 50 – 120 par l'unique instruction :

CIRCLE (C0, L0), 10 * COULEUR, COULEUR

D'où la deuxième possibilité pour tracer la cible :

```
200 ' CIBLE2
210 C0=160   L0=100   CLS
220 FOR COULEUR=1 TO 7
230   CIRCLE(C0,L0),10*COULEUR,COULEUR
240 NEXT COULEUR
```

Pour ne tracer que la deuxième cible il suffit d'effacer l'écran puis d'exécuter le programme à partir de la ligne 200 :

RAZ

RUN 200

Quelle différence dans la rapidité d'exécution ! Une instruction est toujours beaucoup plus efficace que le programme qu'elle remplace : le même effet s'est produit dans le tracé d'une droite point par point (avec une série d'instructions PSET) ou directement avec l'instruction LINE.
Si vous vous sentez plus proche des animaux (nocturnes) que des tireurs, tracez les deux cibles l'une à côté de l'autre en modifiant les coordonnées des centres :

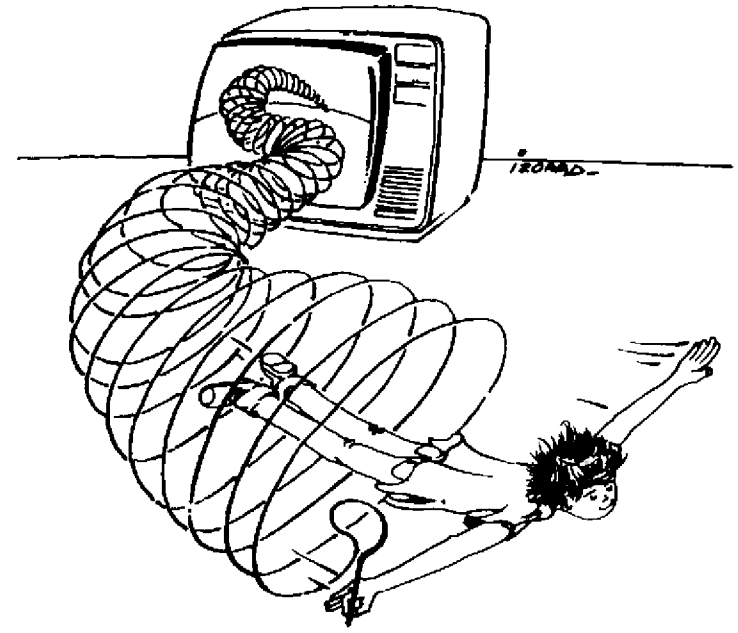
```
 30 C0 =  80 : L0 = 100
210 C0 = 240 : L0 = 100
```


Pour tracer les deux yeux avec l'instruction CIRCLE, placez le tracé de la cible dans un sous-programme que le programme principal appellera une fois pour chaque centre :

```
10  ' HIBOU
20  CLS:SCREEN 7,0,0
30  C0=80   : GOSUB 100
40  C0=240  : GOSUB 100
50  END
100 ' OEIL
110 FOR COULEUR=1 TO 7
120   CIRCLE(C0,100),10*COULEUR,COULEUR
130 NEXT COULEUR
140 RETURN
```

Des cercles à volonté

Comment dessiner des cercles à l'écran en montrant simplement du doigt, par photostyle interposé, le centre du cercle ?



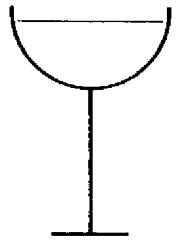
Photostyle ? Ceci est un néologisme (mot nouveau) créé par la très respectable Commission de Terminologie de l'Informatique comme équivalent à "crayon optique". Avec le tableau des préfixes et suffixes grecs et latins présents dans tout dictionnaire, vous pourrez vous livrer au même jeu que cette Commission et faire un petit programme qui crée à volonté des mots qui n'existent pas encore en français (comme par exemple *photophage* = qui mange la lumière !).

Revenons à nos cercles.
C'est l'instruction INPUTPEN C,L(entre parenthèses : pas de parenthèses autour de C,L) qui affecte aux variables C et L les coordonnées du point désigné à l'écran avec de crayon optique. Une fois tracé le premier cercle de centre (C,L) et le rayon choisi (par exemple 3Ø), il suffit pour tracer un nouveau cercle d'aller reprendre les nouvelles valeurs de C et L correspondant au 2e centre désigné à l'écran, et ainsi de suite...

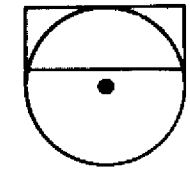
```
10 ' COCERCLE
20 CLS : SCREEN 0,7,7
30 INPUTPEN C,L
40 IF C<0 OR L<0 THEN GOTO 30
50 CIRCLE(C,L),30
60 GOTO 30
```

Un demi-cercle pour un ballon

Si vous avez soif, nous allons d'abord créer le récipient, à savoir un verre ballon, et nous verrons ensuite comment le remplir.
Un verre ballon, c'est à peu près ceci :



Donc un demi-cercle et trois traits. Mais l'instruction CIRCLE donne un cercle entier. Comment n'en garder que la moitié ? Nous effacerons en fait un peu moins de la moitié du cercle avec un rectangle "plein" (dans la couleur du fond). Puisqu'il s'agit de couleur du fond (bleu clair), la boîte sera remplie en couleur négative (– 7) : la couleur 6 sert à dessiner des formes en bleu clair tandis que la couleur – 7 sert à créer un fond bleu clair (en particulier pour effacer).



Pour pouvoir modifier la place du verre à l'écran, plaçons les coordonnées du centre du cercle dans des variables C et L.

```
10 ' VERRE
20 CLS : SCREEN 4,6,6
30 C=160 : L=100
40 CIRCLE(C,L),30,1
50 BOXF(C-30,L-30)-(C+30,L-5),-7
60 LINE(C-30,L)-(C+30,L),1
70 LINE(C,L+30)-(C,L+70),1
80 LINE(C-20,L+70)-(C+20,L+70),1
```

Remplir le ballon... de rouge (PAINT)

Ajoutez la ligne suivante au programme précédent :

90 PAINT (C, L + 15), 1

Êtes-vous satisfait ?
L'instruction PAINT permet de remplir avec une couleur donnée, une courbe fermée *tracée avec la même couleur*. Il suffit de donner les coordonnées d'un point à l'intérieur de la courbe. Le remplissage se fait d'abord du haut vers le bas à partir du niveau du point donné, puis du bas vers le haut pour finir le remplissage à partir du niveau de départ.
Ici pour que le remplissage se fasse du bas vers le haut à partir du fond du verre (ce qui est logique pour un liquide), il faut partir du point (C,L + 29) :

90 PAINT (C,L + 29), 1

Que se passe-t-il si vous faites commencer le remplissage à partir d'un point situé à l'extérieur du verre ? Modifiez la ligne 90 :

90 PAINT (C,L), 1

C'est une marée rouge qui envahit petit à petit l'écran jusqu'à ne laisser qu'un demi-cercle de ciel.
Ceci se produit également lorsque la figure à remplir n'est pas "bien fermée" : une ouverture d'un seul point et inexorablement le flot se répand.
Inutile dans ce cas d'attendre que tout l'écran soit rempli : exécutez à nouveau le programme précédent et appuyez sur la touche INITIALISATION avant la fin du remplissage de l'écran.
Le menu vous est à nouveau proposé et l'appui sur la touche [1] vous rend la main avec votre programme :

LIST

Rien n'est perdu ! C'est ce que l'on appelle un "redémarrage à chaud" (peut-être parce que l'on a eu chaud ?).
Par contre, rien ne se passe si l'on tente de peindre une ligne déjà peinte (le démarrage d'un remplissage à partir d'un point situé sur la ligne est inopérant).
Sur MO 5, vous pouvez également arrêter le remplissage de PAINT en appuyant sur [STOP] ou [CNT][C].
Si, tenant à votre santé, vous préférez la menthe à l'eau, il ne suffit pas de remplir le verre en vert en ne modifiant que la couleur de remplissage ligne 90,

90 PAINT (C,L + 29), 2
RUN

c'est encore une marée, mais verte cette fois, qui envahit l'écran (en attendant la prochaine... noire).
Il faut que *tout le pourtour* de la figure soit de la même couleur (ici 2).

```
10 ' MENTHE
20 CLS : SCREEN 4,6,6
30 C=160 : L=100
40 CIRCLE(C,L),30,2
50 BOXF(C-30,L-30)-(C+30,L-5),-7
60 LINE(C-30,L)-(C+30,L),2
70 LINE(C,L+30)-(C,L+70),2
80 LINE(C-20,L+70)-(C+20,L+70),2
90 PAINT (C,L+29),2
```

Des cercles pleins

Il est très facile de modifier le programme COCERCLE pour qu'il trace des cercles pleins dans une couleur donnée. Mais il serait mieux encore de laisser l'utilisateur choisir la couleur de chaque cercle. On peut imaginer préciser la couleur choisie en tapant le chiffre correspondant au clavier. Mais nous ne voulons pas qu'apparaissent à l'écran les chiffres tapés au clavier (ce qu'on appelle l'"écho") comme cela se passe avec l'instruction INPUT.
Un de ses confrères agit avec plus de discrétion et de rapidité pour entrer des réponses au clavier, lorsque l'on sait à l'avance le nombre de caractères attendus dans la réponse : INPUT$ ( ).
Si par exemple vous attendez une réponse OUI ou NON (3 caractères), il suffit d'écrire :

10 R$ = INPUT$(3)
RUN

Il n'y a pas de point d'interrogation à l'écran. Tapez OUI ou NON (inutile ici d'appuyer sur la touche ENTRÉE).
Rien ne s'affiche à l'écran. Mais vous pouvez vérifiez ce que contient la variable R$:

PRINT R$

Notez bien que la réponse affectée à une variable par INPUT$ est toujours considérée comme une "chaîne de caractères".

Pour colorier les cercles la réponse attendue est une couleur précisée par son numéro, donc un seul caractère.
Mais attention ! Si les chiffres attendus 1, 2, 3... sont enregistrés par l'intermédiaire de la fonction INPUT$, malgré leur apparence ils ne seront pas considérés comme des nombres par l'ordinateur, mais comme des chaînes de caractères. Il sera impossible de faire le moindre calcul avec eux.
Ce sont des *chaînes de caractères* formées d'un seul caractère, qui est un chiffre. Ça se complique !
Pour faire des calculs, il faut des nombres, des vrais, provenant de variables *numériques*.

C'est la même chose dans l'instruction PAINT : la couleur doit être une variable (ou une constante) *numérique*.
Heureusement il est possible de "convertir" (ce qui ne se verra qu'à l'intérieur de l'ordinateur) une chaîne de caractères formée par des chiffres en un nombre.
C'est ce que fait la fonction VAL (pour "valeur numérique de"...)

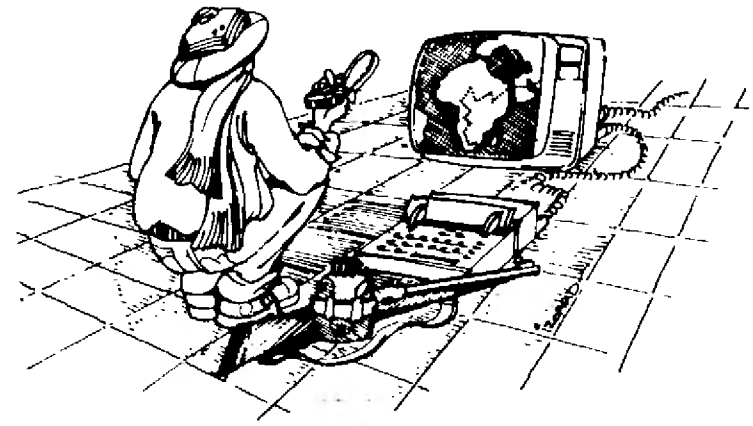
COUL = VAL (COUL$)

convertit la chaîne de caractères COUL$ (à condition que celle-ci ait toutes les apparences d'un nombre) en nombre.
La variable numérique COUL peut maintenant être utilisée dans l'instruction PAINT et vous pouvez rédiger le programme qui permettra à l'utilisateur de dessiner des cercles pleins dans la couleur de son choix avec le crayon optique :

```
10 'CERCLEF
20 CLS : SCREEN ,6,6
30 COUL$=INPUT$(1)
40 COUL=VAL(COUL$)
50 INPUTPEN C,L
60 IF C<0 OR L<0 THEN GOTO 50
70 CIRCLE(C,L),20,COUL
80 PAINT(C,L),COUL
90 GOTO 30
```

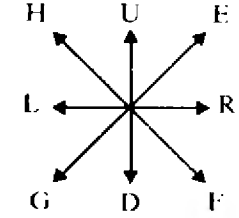
Si vous essayez tous les chiffres de 0 à 7, il y en a nécessairement un qui donnera un cercle invisible.

Dessiner à la boussole (DRAW)



Assez de ronds.
Pour les carrés et les rectangles, BOX remplit parfaitement son office.
Pour tous les dessins formés de segments de droites, c'est LINE que nous avons utilisé. Mais cela suppose de calculer d'abord les coordonnées de tous les points du dessin, en ayant recours éventuellement à un certain Pythagore.
Au lieu de dire à chaque fois à l'ordinateur "Joins tel point à tel point", on peut lui donner une boussole et lui dire à la fois la direction à prendre et la longueur dont il faut se déplacer dans cette direction.
C'est ce que fait (entre autres choses) l'instruction DRAW ("tracer").

Il y a huit directions possibles :



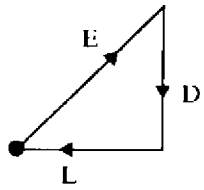
U pour Up (vers le haut)
R pour Right (vers la droite)
D pour Down (vers le bas)
L pour Left (vers la gauche)
E, F, G, H pour les directions intermédiaires (à 45° par rapport aux précédentes)

Essayez :
DRAW "U30"

L'instruction DRAW est suivie d'une chaîne de caractères, ici la plus simple possible : une direction (U) suivie d'une longueur (30). En l'absence d'autre précision, le point de départ est le centre de l'écran (160, 100)

DRAW "E30D30L30"

dessine un triangle dont le coin inférieur gauche et au centre de l'écran.



Il y a peut-être quelque chose qui vous gêne : les trois directions E, D, L sont suivies du même nombre 30 et manifestement le segment à 45° (E) est plus long que les deux autres !
Il en va de même pour les quatre directions, E, F, G, H : elles sont suivies, non pas de la longueur à parcourir, mais de la longueur correspondante sur une horizontale ou une verticale. Ainsi il est inutile de calculer la longueur de l'hypothénuse : Pythagore est passé à l'intérieur de la machine.
Pour placer le point courant en un point quelconque de l'écran, il suffit de lui dire M (Move = déplacer) suivi des coordonnées du point :

RAZ

DRAW "M250,100E30D30L30"

Les commandes à l'intérieur des guillemets peuvent être séparées par des points-virgules ou des espaces :

DRAW "M250,100;E30;D30;L30"

Si vous voulez lever le crayon en cours de dessin, il suffit de faire précéder la commande correspondante par B (Blank pour blanc) :

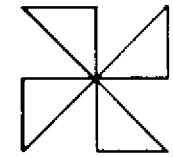
DRAW "BM250,100;E30;D30;L30;

Ainsi pour tracer un trait pointillé horizontal :

```
10 FOR I = 1 TO 10
20 DRAW "R8BR8"
30 NEXT I
```


Un moulinet en couleurs

Chaque aile sera d'une couleur différente. Mais commencez par tracer les quatre ailes de la même couleur (blanc sur fond noir).



```
10 ' MOULIN
20 CLS : SCREEN 7,0,0
30 DRAW "E30D30L30"
40 DRAW "F30L30U30"
50 DRAW "G30U30R30"
60 DRAW "H30R30D30"
```

Pour tracer les ailes en rouge (1), il suffit d'ajouter C1 (C pour Color) devant la suite de commandes de la première aile.

```
30 DRAW "C1;E30D30L30"
```

RUN

Le point-virgule entre la commande C1 et les autres commandes est facultatif.
Traçons maintenant les quatre ailes dans des couleurs différentes :

```
10 ' MOULIN
20 CLS : SCREEN 7,0,0
30 DRAW "C1;E30D30L30"
40 DRAW "C2;F30L30U30"
50 DRAW "C3;G30U30R30"
60 DRAW "C4;H30R30D30"
```

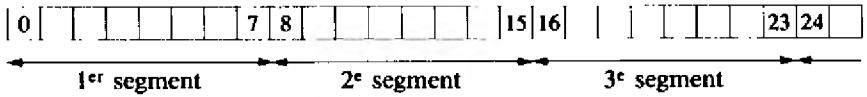
Au centre du moulinet, les couleurs se chevauchent un peu : l'aile bleue déteint sur la rouge. Vous pouvez observer tranquillement ce qui se passe au cours du tracé en plaçant une apostrophe devant la quatrième instruction DRAW (la ligne 60 n'est pas exécutée dans ce cas).

60 'DRAW "C4;H30R30D30"

De même pour la ligne 50.

Petit intermède sur l'affichage des couleurs à l'écran

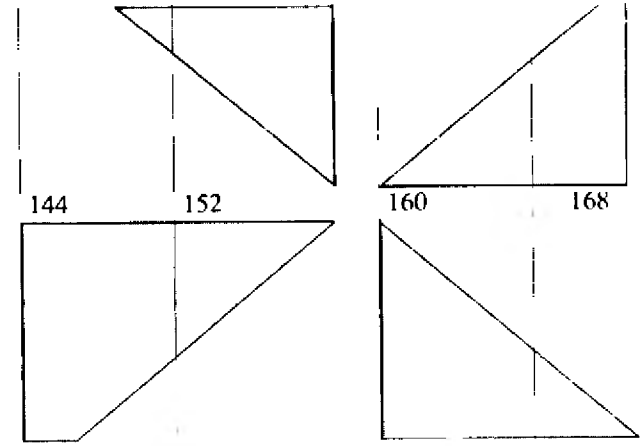
Sans doute avez-vous déjà rencontré ce chevauchement de couleurs. Il vient de la manière dont les couleurs sont affichées à l'écran. Sur une ligne verticale vous pouvez allumer tous les points successifs avec des couleurs différentes : tracez (avec l'instruction PSET) une série de sept points de 7 couleurs différentes. Inutile de chercher une loupe, vous atteignez la limite de la définition de l'écran.
Sur une ligne horizontale il n'en va pas de même. Chaque ligne est divisée en segments successifs de huit points, qui correspondent à la largeur d'un caractère :



Sur chaque segment, on ne peut utiliser qu'une seule couleur (autre que celle du fond) pour dessiner ou écrire, et inversement si la couleur d'écriture est choisie, on ne peut avoir qu'une couleur de fond.
Si, dans un même segment, on allume plusieurs points avec des couleurs différentes, c'est la dernière couleur demandée qui l'emporte pour tous les points allumés du segment (sauf ceux qui sont dans la couleur du fond).
Vous pouvez aussi le vérifier avec une suite d'instructions PSET. Ainsi dans le moulinet, le dernier triangle tracé (le bleu dans la version intégrale du programme) colorie en bleu les points rouges qui appartiennent à des segments où sont tracés ensuite des points bleus.
Comment améliorer le dessin ?

Lever le crayon en cours de dessin

Ce peut être une solution à notre problème. Pour éviter le chevauchement des couleurs, il ne faut pas avoir deux couleurs différentes dans un même segment.
Le centre du moulinet est au centre de l'écran (on n'a pas précisé d'autre point de départ), donc en colonne 160, c'est-à-dire sur le premier point du segment 160-167.



Si nous décalons, comme sur la figure, la deuxième aile d'un point vers le bas (BD1 = Blank Down 1), aucun de ses points ne sera sur un segment de la première aile :

```
30 DRAW "C1;E30D30L30BD1"
```

L'effet est très discret, pour ne pas dire invisible.
Décalons de même la troisième aile vers la gauche (BL1) et la dernière vers le haut (BU1).
Levons donc le crayon pour un point entre chaque aile :

```
10 ' MOULIN
20 CLS : SCREEN 7,0,0
30 DRAW "C1;E30D30L30;BD1"
40 DRAW "C2;F30L30U30;BL1"
50 DRAW "C3;G30U30R30;BU1"
60 DRAW "C4;H30R30D30"
```

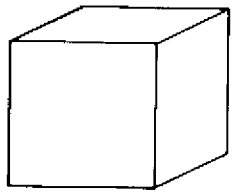
Cette fois, les couleurs sont bien respectées.
Vous pouvez même peindre l'intérieur de chaque aile :

```
10 ' MOULIN
20 CLS : SCREEN 7,0,0
30 DRAW "C1;E30D30L30;BD1"
35 PAINT (180,90),1
40 DRAW "C2;F30L30U30;BL1"
45 PAINT (170,120),2
50 DRAW "C3;G30U30R30;BU1"
55 PAINT (140,110),3
60 DRAW "C4;H30R30D30"
65 PAINT (150,90),4
```

Changer d'échelle

Pour dessiner un cube en perspective, plusieurs solutions sont possibles. Trouvez la vôtre.
Inutile de chercher un tracé qui ne repasse jamais deux fois sur le même segment, c'est impossible ! Par contre il est possible de ne repasser deux fois que sur une arête.

Notez votre point de départ.



Voici notre solution :

```
10 'CUBE
20 CLS
30 C$="R40D40E20U40L40G20D40R40U40E20"
40 DRAW C$
```

Nous avons placé la chaîne de caractères qui suit l'instruction DRAW dans une variable pour ne pas avoir à la réécrire ultérieurement.
Et maintenant, changeons d'échelle !
Pour obtenir ce cube avec une arête quatre fois plus petite, il suffit de rajouter S1 (Scale = échelle) au début de la suite de commandes qui trace le cube :

```
40 DRAW "S1" + C$
```

Vous avez 63 échelles différentes à votre disposition :

S1 échelle 1/4
S2 échelle 2/4 ou 1/2
S3 échelle 3/4
S4 échelle 4/4 ou 1
S5 échelle 5/4
.....
S8 échelle 8/4 ou 2
.....
S12 échelle 12/4 ou 3
.....
S63 échelle 63/4 ou 15,75

L'échelle 1 équivalente à S4 est celle qu'on obtient en l'absence de commande S.

Pour doubler les dimensions du cube, changez le "1" en "8" :

```
40 DRAW "S8" + C$
```

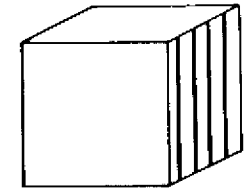
Un point reste commun à tous ces cubes : le point de départ du dessin du cube.
Que se passe-t-il lors du changement d'échelle si ce point de départ n'est pas le centre de l'écran ?
En revenant à l'échelle 1 (c'est-à-dire sous-entendu S4), modifiez le point de départ du tracé :

```
40 DRAW "BM160,50" + C$
```

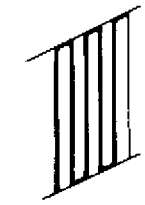
puis doublez l'échelle :

```
40 DRAW "S8;BM160,150" + C$
```

Les changements d'échelle ne s'appliquent pas au MOVE absolu.
Est-il possible d'augmenter la sensation de volume ? Pour donner l'impression de relief, on peut jouer sur les couleurs – regardez les graphismes des magazines, les logos publicitaires...
Mais même avec une seule couleur, on peut se débrouiller, c'est ce que font les graphistes pour les revues tirées dans une seule couleur.
Regardez ce cube et comparez-le au précédent :



Ajoutez à votre programme les instructions pour obtenir la face avant en couleur et la face latérale hachurée de rayures parallèles.



```
10 'CUBE
20 CLS
30 DRAW "R40D40E20U40L40G20D40R40U40E20"
40 PAINT(180,120)
50 FOR I=1 TO 5
60    DRAW"G2D40G2U40"
70 NEXT I
```

Vous pouvez régler l'écartement entre les rayures verticales pour obtenir la sensation de grisé désirée.
Modifions l'échelle, par exemple diminuons-la de moitié.

```
10 'CUBE
20 CLS
25 DRAW "S2"
30 C$="R40D40E20U40L40G20D40R40U40E20"
40 DRAW C$
50 PAINT(180,120)
60 FORI=1 TO 5
70   DRAW"G2D40G2U40"
80 NEXT I
```

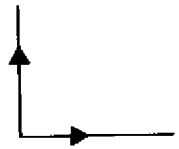
Tracer en aller-retour

Voici une étoile, ou plutôt la scintillation produite par une étoile, car les étoiles sont comme le soleil : rondes ! Mais même avec le télescope le plus puissant du monde, vous ne verrez jamais que la scintillation des étoiles. Elles seront plus lumineuses et donc deviendront visibles, mais toujours par leur scintillation.
D'abord une seule étoile : en partant du centre, il y a huit branches à tracer en aller-retour. Pour vous éviter de tracer l'aller *et* le retour, il suffit de faire précéder la commande correspondante par la lettre N (pour "No update" = sans mise à jour des coordonnées courantes). Après le tracé demandé le point courant est à nouveau le point de départ.
Essayez :

```
30 DRAW "NU6"
```

Apparemment, vous auriez eu le même effet avec "U6".
Essayez alors :

```
30 DRAW "NU6NR6"
```



Le point traceur trace U6, et repart de l'origine pour tracer R6. D'où l'étoile :

```
10 ' ETOILES
20 CLS
30 A$="NU6NR6ND6NL6NE4NF4NG4NH4"
40 DRAW A$
```

Nous avons placé la chaîne de caractères formée de la suite des commandes dans une variable afin de pouvoir la réutiliser plus facilement.
Une deuxième étoile dans le coin ? par exemple à 20 points à droite et 10 points au-dessus.
Précisons le déplacement à effectuer à partir du dernier point tracé à l'écran (le centre de la première étoile) par "BM + 20, – 10".
Les signes + et – signalent à l'ordinateur que les nombres qui suivent indiquent un déplacement "relatif" et ne sont pas les *coordonnées* d'un point à joindre (déplacement "absolu") :

```
50 DRAW "BM + 20, – 10" + A$
```

A partir du centre de cette deuxième étoile (dernier point tracé) ajoutons-en une troisième :

```
60 DRAW "BM + 0, + 30" + A$
```

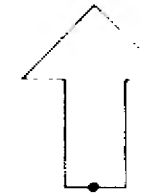
Remarquez que même le zéro doit être précédé d'un signe (+ ou –, vous avez le choix !). Nous pouvons déplacer l'ensemble de notre petit coin de ciel au point (200,50), il suffit de rajouter au début du programme un déplacement absolu.

```
2 DRAW "BM200,50"
```

Tourner le dessin d'un quart de tour

La flèche ci-contre ne comporte que des traits à 90° ou 45°.
Dessinez-la à l'écran en plaçant le point • au centre de l'écran.
Une nouvelle possibilité de l'instruction DRAW est de faire tourner une figure d'un quart de tour, un demi-tour, trois quarts de tour ou revenir au sens normal :

A1	1/4 tour vers la droite
A2	1/2 tour vers la droite
A3	3/4 tour vers la droite
A0	revenir au sens normal



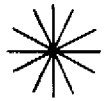
Essayez ces commandes sur la flèche :

```
10 ' FLECHE
20 CLS
30 F$="R5U30R15H20G20R15D30R5"
40 DRAW F$
50 DRAW"A1"+F$
60 DRAW"A2"+F$
70 DRAW"A3"+F$
80 DRAW "A0"
```

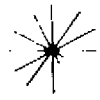
Pour voir le rôle de chaque ligne, placez une apostrophe devant l'instruction à supprimer provisoirement.

Une étoile qui scintille encore plus

Vous pensez peut-être que cette instruction DRAW est bien pratique mais restreinte aux dessins comportant uniquement des segments horizontaux, verticaux ou à 45° avec l'horizontale ou la verticale.
Il n'en est rien : à partir d'un point quelconque du dessin, il est possible de tracer un trait dans n'importe quelle direction. Par exemple voici une étoile qui scintille un peu plus que la précédente grâce à des rayons d'inclinaisons différentes de 45 ° :


l'étoile

"un quart" d'étoile



Il suffira d'en dessiner "un quart" et de faire tourner ce quart, comme la flèche dans le cas précédent, d'1/4 tour, 1/2 tour et 3/4 de tour pour reconstituer l'étoile.
Pour tracer le premier quart, utilisons le déplacement relatif qui permet de tracer des traits d'inclinaison quelconque.
Après quelques essais avec diverses valeurs (ou un dessin sur papier quadrillé), nous choisissons :

```
10 ' ETOILE2
20 CLS
30 Q$="NU6;NM+5,-3;NM+3,-5"
40 DRAW Q$
```


Puis pour obtenir l'étoile entière :

```
50 DRAW "A1"+Q$
60 DRAW "A2"+Q$
70 DRAW "A3"+Q$
80 DRAW "A0"
```

Ayant réalisé ce dessin, vous aurez peut-être envie de l'utiliser dans diverses occasions : par exemple dans des programmes différents ou bien au cours d'un même programme en le plaçant successivement à divers endroits de l'écran (éventuellement avec le crayon optique).
Pour cela, il serait nécessaire de pouvoir placer les coordonnées du point de départ du dessin dans deux variables (C et L).
Or, l'instruction DRAW n'admet que des chaînes de caractères, que ce soient des constantes (entre guillemets) ou des variables (comme A$ ou Q$).
Pour introduire une variable numérique dans une commande DRAW, il faut la placer dans la chaîne de caractères, précédée du signe = , et suivie d'un point-virgule.
Pour désigner le contenu de la variable C, on écrira donc :

=C;

La commande qui place le point courant en un point de coordonnées C,L se rédige sous la forme :

DRAW "BM=C;,=L;"

Vous pouvez maintenant rédiger le programme qui dessine l'étoile précédente à l'écran en plaçant le centre de l'étoile avec le crayon optique (n'oubliez pas la virgule qui sépare colonne et ligne dans la commande BM).
Il suffit de rajouter entre les lignes 30 et 40 du programme "ETOILE2" précédent, les six lignes suivantes :

```
31 SCREEN 0,7,7
32 INPUTPEN C,L
33 IF C<0 THEN 32
34 DRAW "BM=C;,=L;"
```

Si vous n'aimez pas les numéros de lignes trop serrés, allez voir plus loin la commande RENUM.

Des DRAW les uns à l'intérieur des autres

L'ensemble des commandes élémentaires disponibles avec l'instruction DRAW constitue un langage graphique autonome. Ce langage possède même une commande qui rappelle l'utilisation d'un sous-programme en Basic.
Reprenons l'exemple du moulin. Plaçons dans la variable A$ le tracé d'une aile et dans M$ celui du moulin.

```
10 A$="E30D30L30"
20 M$=A$+"A1"+A$+"A2"+A$+"A3"+A$+"A0"
30 DRAW M$
```

La chaîne M$ contient la suite de *toutes* les commandes élémentaires nécessaires pour le tracé du moulin. Vérifiez-le avec :

```
PRINT M$
E30D30L30A1E30D30L30A2E30D30L30A3E30D30L30A0
```

Ouf !
La suite d'instructions "E30D30L30" apparaît quatre fois.
Il est possible d'obtenir le même résultat de façon plus simple (et en économisant de la place en mémoire) en demandant directement *le tracé* de l'aile à chaque fois qu'il intervient.
De même dans un programme, on "appelle" un sous-programme (destiné à une opération déterminée) à chaque fois qu'il faut faire cette opération. Cela évite de réécrire à chaque fois toutes les lignes du sous-programme.
En outre, les sous-programmes permettent de structurer un programme en le décomposant en plusieurs parties indépendantes et réutilisables en différentes occasions.
Ici il est possible d'"appeler" le tracé d'une chaîne donnée à l'intérieur de l'instruction DRAW. Pour le moulin cela s'écrit :

```
10 A$="E30D30L30"
20 MOULIN$="XA$;A1XA$;A2XA$;A3XA$;A0"
30 DRAW MOULIN$
```

La lettre X placée devant A$ signifie : DRAW A$ (le point-virgule après XA$ est obligatoire).
Pour voir la différence entre M$ et MOULIN$, faites :

```
PRINT MOULIN$
XA$ ; A1XA$ ; A2XA$ ; A3XA$ ; A0
```

C'est un peu plus court que M$ et en tout cas plus lisible.
Vous disposez maintenant de tout ce qu'il faut pour créer vos dessins

à partir de ce langage graphique. Chaque dessin est une chaîne de caractères comportant les ordres élémentaires et qui peut se décomposer en sous-dessins distincts, chacun correspondant à une chaîne de caractères. L'ensemble peut alors être stocké dans un fichier de données, comme nous le verrons aux chapitres IV et V. Vous pourrez également vous essayer à les modifier avant chaque affichage pour créer une "animation".

Mémoriser et déplacer une image (GET, PUT)

Il est des jours où l'on aimerait conserver les compositions graphiques que l'on vient d'effectuer à l'écran, que ce soit pour des raisons utilitaires ou esthétiques. Jusque-là, nous n'avions comme solution que de programmer ces dessins ou de noter soigneusement dans des variables tous les points, lignes, boîtes, cercles, couleurs,... utilisés.
Vous disposez maintenant d'une nouvelle instruction qui permet de mémoriser une partie de l'image, quel que soit son contenu, dans un tableau numérique.
Reprenons les trois étoiles tracées en aller-retour :

```
10 ' ETOILES
20 CLS
30 A$="NU6NR6ND6NL6NE4NF4NG4NH4"
40 DRAW A$
50 DRAW "BM+20,-10"+A$
60 DRAW "BM+0,+30"+A$
```

Nous allons les conserver ensemble dans un tableau. Elles sont entièrement contenues dans le rectangle de sommets opposés : (150,80) – (190,140). On pourrait même les inscrire dans un rectangle plus petit mais cela ne faciliterait pas les calculs.
L'instruction GET permet de mémoriser ce rectangle dans un tableau qu'il va falloir déclarer.
La dimension du tableau va se calculer de la manière suivante :

— le rectangle (150,80) – (190,140) contient 41 points en colonne et 61 en ligne, soit 2501 points exactement. Attention aux bornes, les points extrêmes font partie du rectangle !
— Il faut savoir que sur TO 7, chaque point à mémoriser occupe en mémoire la moitié d'un octet. On met donc deux points par octet. D'autre part dans un tableau numérique ordinaire, en simple précision, chaque élément du tableau occupe 4 octets. Donc un élément du tableau peut contenir huit points.

— En conséquence, la dimension du tableau est au moins de 2501 divisé par 8, soit 312.625, ce qui ne tombe pas juste. Nous réserverons donc un tableau de 313 éléments, ce qui s'écrit :

```
80 DIM T(312)
```

Sachant que la numérotation des éléments d'un tableau commence toujours à 0.
La mémorisation de la portion d'image va s'écrire :

```
70 ' MEMORISATION
80 DIM T(312)
90 GET(150,80)-(190,140),T
```

Sur MO 5, chaque point à mémoriser occupe 1 octet. Il faut donc utiliser un tableau deux fois plus grand :
80 DIM T (625).
Remarquons que s'il n'est pas gênant de déclarer un tableau un peu trop grand, l'instruction GET refuse de mémoriser si le tableau est trop petit : un message OM Error ou Error 7 apparaît immédiatement.
La restitution de l'image dans une partie différente de l'écran se fait alors par l'instruction PUT, de syntaxe analogue à la précédente. Le rectangle défini dans PUT doit être exactement de mêmes dimensions que celui utilisé dans GET, sinon un message OM Error ou Error 7 apparaît, ou bien on perd complètement l'image à l'écran.

```
100 ' DEPLACEMENT
110 CLS
120 PUT (50,50)-(90,110),T
130 PUT (250,50)-(290,110),T
```

Cette paire d'instructions (PUT, GET) vous permettra de constituer une petite banque d'images, pour composer, tout au long d'un programme, des écrans aux agencements modifiables.
Une dernière remarque : ces deux instructions ne doivent pas être confondues avec deux instructions de même nom (PUT et GET) destinées aux fichiers à accès direct que nous verrons au chapitre V.

Autres compléments (HEX$, SPACE$, STRING$, DEF FN, DEFUSR, MID$, SWAP, AUTO, RENUM, SEARCH)

De nouveaux objets BASIC qui ne concernent pas le graphique, viennent s'ajouter au contenu de la mémoire morte, pourtant déjà bien remplie !
Pour le MO 5, il s'agit d'abord d'une fonction (HEX$), de deux instructions (MID$ et DEF USR) et d'une commande (AUTO).

Celles-ci existent déjà sur la cartouche BASIC du TO 7.
S'y ajoute une commande particulière à MO 5 : SEARCH.
Pour les deux machines MO 5 et TO 7, s'ajoutent en plus deux fonctions (SPACE$ et STRING$), deux instructions (SWAP et DEF FN) et une commande RENUM.

Nous allons toutes les passer en revue, en commençant par les fonctions.

HEX$ (I)

donne une chaîne de caractères représentant le nombre I en hexadécimal, c'est-à-dire en base 16.

```
PRINT HEX$ (16)
10
```

Cette fonction est très pratique pour afficher les adresses et le contenu des octets de la mémoire de votre ordinateur. Encore faut-il lire couramment l'hexadécimal !

SPACE$ (I)
forme une chaîne de caractères composée de I blancs. Par exemple :

```
PRINT SPACES (120)
```

efface 3 lignes.

STRING$ (I,"A")
forme une chaîne de caractères composée de I fois le caractère A.
Le caractère peut être indiqué par une constante ("A"), une variable (A$), ou par son code en ASCII (65).
Exemple :

```
10 FOR I=1 TO 23
20   PRINT STRING$(10,".")
30 NEXT I
```

DEF FN
Il vous est peut-être arrivé de faire 10 ou 20 fois le même calcul avec des variables différentes dans le même programme.
DEF FN... va vous permettre de définir une fonction qui effectue ce calcul. Supposons que vous fassiez plusieurs fois la moyenne de trois nombres ; vous définirez en début de programme :

```
DEF FNMOY (A,B,C) = (A + B + C)/3
```

Pour calculer la moyenne de trois nombres, vous utiliserez cette fonction comme une fonction du BASIC : par exemple

```
PRINT FNMOY (123, 456, 789)
```

ou pour la moyenne de trois variables X, Y, Z :
PRINT FNMOY (X,Y,Z)

La définition de la fonction ne peut s'écrire que sur une seule ligne mais avec autant de variables que vous le souhaitez. Évidemment, il faudra mettre autant de variables au moment de l'appel de la fonction.
Il est également possible de définir des fonctions chaînes de caractères. Ainsi pour convertir un nombre en une chaîne, mais sans le signe ou l'espace en premier, vous définirez :

DEF FNCH$ (I) = MID$ (STR$ (I), 2)

N'oubliez pas que le nom de la fonction que vous définissez commence toujours par FN. La suite du nom obéit aux même règles que les noms de variables.

Exemple :

```
10 ' CALCUL DES POINTS D'UNE DROITE
20 DEF FNDROITE(X)=3*X+2
30 FOR I=0 TO 10
40    Y(I)=FNDROITE(I)
50 PRINT I,Y(I)
60 NEXT I
```

DEF USR
Cette instruction permet de définir des fonctions utilisateur écrites en langage machine, et de les appeler par USR en passant un paramètre. Le détail sur ces fonctions est donné en Annexe 3.

MID$ (instruction)
Nous avons déjà vu, à propos du BASIC en mémoire morte, la fonction MID$ ( ) qui permet d'extraire un morceau d'une chaîne de caractères. Il existe aussi une instruction qui s'écrit également MID$ ( ), destinée à faciliter la modification de chaînes de caractères, quand cette modification ne change pas la longueur de la chaîne.

Exemple :

```
A$ = "AUTOMATISME"
MID$ (A$,8,2) = "QU"
PRINT A$
AUTOMATIQUE
```

Dans cette instruction, le 1er nombre indique l'endroit où doit commencer la modification, le second en précise la longueur. La chaîne qui va remplacer la partie indiquée s'écrit à droite du signe = . Son résultat est identique à celui qu'on aurait obtenu en écrivant :

A$ = MID$ (A$, 1,7) + "QU" + MID$ (A$, 10)

mais comme vous le constaterez vous-même, la première forme est plus "parlante", et aussi plus rapide.

SWAP
Cette instruction permet d'échanger le contenu de deux variables :
SWAP A,B échange les contenus des variables numérique A et B,
SWAP R$, P$ échange les contenus des chaînes de caractères R$ et P$.

```
10 A$="BLANC"
20 B$="BONNET"
30 PRINT A$,B$
40 SWAP A$,B$
50 PRINT A$,B$
```

Son utilisation est particulièrement appréciée dans le tri des éléments d'un tableau.

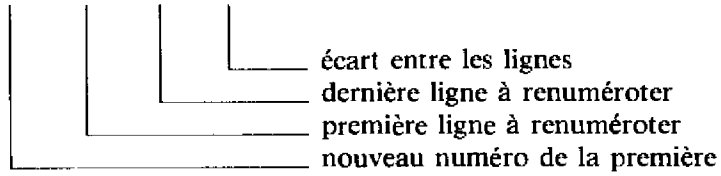
AUTO
Les bons programmeurs, dont vous faites certainement partie, ceux qui écrivent leur programme entièrement sur papier avant de le taper au clavier, trouvent un peu fastidieux d'avoir à écrire systématiquement tous les numéros de ligne.
La commande AUTO vous en dispense, en composant automatiquement tous ces numéros. Il suffit de lui donner deux nombres : le numéro de la première ligne et le nombre qui sert d'écart entre deux lignes.

AUTO 500, 5 propose automatiquement les numéros de lignes commençant en 500 et espacés de 5.
Le passage d'une ligne à la suivante se fait au moment où l'on appuie sur [ENTREE]. L'arrêt de la numérotation automatique est obtenu par [CNT][C].
Les deux nombres ne sont pas obligatoires. En leur absence, c'est la valeur 10 qui est prise par défaut pour chacun d'eux.

RENUM
Cette commande très utile permet de renuméroter tout ou partie d'un programme résidant en mémoire. Dans sa forme la plus simple :
RENUM renumérote de 10 en 10 à partir de la 1re ligne.
Il est possible de faire cette opération à partir d'une ligne quelconque du programme, et avec un écartement quelconque entre les lignes.

Prenons par exemple le programme BONHOMME ANIMÉ (voir Initiation au Basic TO 7). Nous pouvons "éloigner" le sous-programme qui commence à la ligne 2000 et le mettre à la ligne 5000 avec un écart de 100 en 100 :

RENUM 5000, 2000, 2020, 100




```
10   ' BONHOMME ANIME
20   CLS : SCREEN 0,6,6
30   BOX (95,100)-(105,110)
40   LINE (100,110)-(100,140)
50   LINE (100,140)-(80,170)
60   LINE (100,140)-(120,170)
70   '
80   C=0 :GOSUB 1000
90   FOR I=1 TO 100 : NEXT I
100  C=6 : GOSUB 1000
110  C=0 : GOSUB 2000
120  FOR I=1 TO 100 : NEXT I
130  C=6 : GOSUB 2000
140  GOTO 80
999  '
1000 LINE (100,120)-(70,100),C
1010 LINE (100,120)-(130,100),C
1020 RETURN
1999 '
2000 LINE (100,120)-(70,120),C
2010 LINE (100,120)-(130,120),C
2020 RETURN
```

Ceci conduit à :

```
10   ' BONHOMME ANIME
20   CLS : SCREEN 0,6,6
30   BOX (95,100)-(105,110)
40   LINE (100,110)-(100,140)
50   LINE (100,140)-(80,170)
60   LINE (100,140)-(120,170)
70   '
80   C=0 :GOSUB 1000
90   FOR I=1 TO 100 : NEXT I
100  C=6 : GOSUB 1000
110  C=0 : GOSUB 5000
120  FOR I=1 TO 100 : NEXT I
130  C=6 : GOSUB 5000
140  GOTO 80
999  '
1000 LINE (100,120)-(70,100),C
1010 LINE (100,120)-(130,100),C
1020 RETURN
1999 '
5000 LINE (100,120)-(70,120),C
5100 LINE (100,120)-(130,120),C
5200 RETURN
```

Remarquez que les branchements indiqués aux lignes 110 et 130 sont modifiés en fonction de la nouvelle numérotation. C'est pratique, non ? Comme la rénumérotation va jusqu'à la fin du programme, il n'était pas utile de le préciser, on aurait pu écrire pour obtenir le même résultat :

RENUM 5000, 2000,, 100

La partie à renuméroter peut se trouver en milieu de programme. Ainsi peut-on renuméroter le premier sous-programme :

RENUM 1500, 1000, 1020, 20

Toutefois, cette renumérotation ne permet pas de déplacer une partie du programme pour la placer avant, ou après une autre qui n'aurait pas bougé. L'écriture suivante n'est pas autorisée sur la version originale du programme :

RENUM 3000, 1000, 1020, 20

En effet, le 1er sous-programme ne peut pas "sauter" par-dessus le second.
La syntaxe complète de RENUM est celle-ci :

RENUM nouveau n° (1re ligne), ancien n° (1re ligne), ancien n° (dernière ligne), écart entre les lignes.

SEARCH **(Spécial MO 5)**

Cette commande est destinée à faciliter l'exploration d'un programme pour y rechercher les lignes où telle ou telle variable ou constante est

employée ou pour y rechercher telle ou telle instruction (par exemple les GOTO ou les GOSUB).
Cette exploration permet ensuite de documenter, ou de modifier les parties du programme concernées.
Plus généralement, elle affiche sur l'écran, comme LIST, les lignes où se trouve une chaîne de caractères que l'on recherche systématiquement.
Par exemple, le programme étant résidant en mémoire centrale, pour y rechercher la variable NBJ, on écrira :

```
SEARCH "NBJ"
```

Aussitôt, s'affichent toutes les lignes contenant NBJ. Comme pour LIST, la recherche peut ne porter que sur une partie de programme. Pour rechercher NBJ à partir de la ligne 1000, on écrira :

```
SEARCH "NBJ", 1000 -
```

La chaîne de caractères à rechercher peut être une instruction : GOSUB, par exemple, pourra donner une idée de la décomposition du programme en sous-programmes. On peut aussi rechercher les numéros de ligne, ce qui donne la liste de toutes les instructions qui y dirigent, sauf la ligne elle-même, les numéros de chacune des lignes n'étant pas pris en compte dans la recherche.
La commande SEARCH se comporte donc comme un LIST sélectif, très pratique pour la mise au point de programmes.

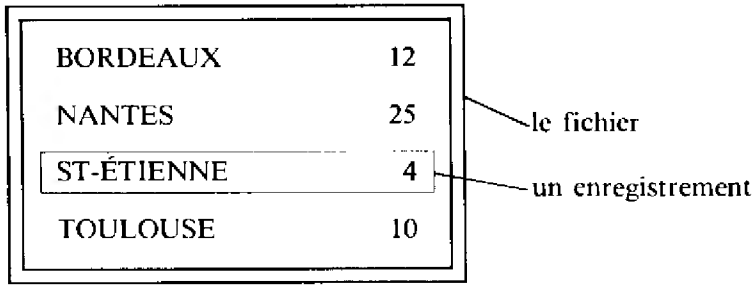
Chapitre IV / Les fichiers séquentiels

Qu'est-ce qu'un fichier de données ?

Les programmes que vous avez enregistrés sur disquette jusqu'à présent constituent la première catégorie de fichiers, celle des "fichiers-programmes". Leur gestion est très facile.
La deuxième (et dernière) catégorie de fichiers est celle des "fichiers de données".
Un fichier de données est constitué d'une liste de mots (chaîne de caractères) et (ou) de nombres : liste de dépenses, de noms, d'adresses, de résultats de jeux...
Il n'y a pas une seule instruction, pas une seule commande dans un fichier de données : rien que des *données*, exactement comme une suite de données dans une instruction DATA.
Stocker ses données dans un fichier plutôt que dans un programme présente plusieurs avantages : d'abord on peut stocker beaucoup plus de données sur disquette, et ensuite un même fichier de données peut servir dans plusieurs programmes différents.
Notons d'abord qu'il est impossible d'enregistrer, de charger et encore moins d'exécuter un fichier de données avec une seule instruction comme on le fait avec un fichier-programme (SAVE, LOAD, RUN).
C'est un paquet inerte.
Pour gérer ce paquet de données : l'enregistrer, le relire à volonté, modifier une des données... vous pouvez utiliser le lecteur de cassettes ou le lecteur de disquettes. Avec le lecteur de disquettes, vous avez à votre disposition deux procédés différents pour gérer les fichiers de données. Avec le lecteur de cassettes, vous ne disposez que du premier procédé.
Dans ce chapitre nous allons voir le premier de ces procédés : il consiste à enregistrer (et relire) toutes les données *les unes à la suite des autres* sur la disquette (ou la cassette). C'est ce qu'on appelle constituer un "fichier à accès séquentiel".
Le deuxième procédé, que nous étudierons dans le prochain chapitre, permet d'enregistrer (et relire) directement n'importe quelle donnée du fichier. Le fichier ainsi constitué est alors un "fichier à accès direct".

La gestion d'un fichier séquentiel

Commençons donc par le type de fichiers le plus simple : les fichiers séquentiels. Si vous voulez conserver la liste de vos amis avec le total de leurs points au fur et à mesure de vos parties de bridge, de belote ou de tarot, comptabiliser le nombre de buts marqués au cours de la saison par toutes les équipes d'une division, surveiller l'évolution des prix en notant le prix de vos café, huile, jus d'orange, lessive habituels... vous aurez une liste dont chaque élément sera constitué d'une chaîne de caractères et d'un nombre :



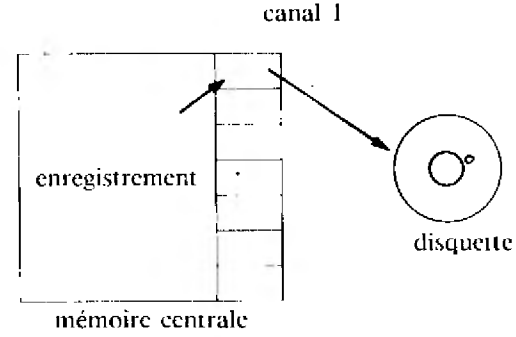
Chacun des éléments constitue un "enregistrement". Un enregistrement peut être plus compliqué. Par exemple le fichier des salariés d'une entreprise, qui sert, entre autres, à établir les bulletins de paie contient, pour chaque employé : les nom, prénoms, adresse, numéro de sécurité sociale salaire brut, retenues...
Un fichier de recettes de cuisine contient : la liste des ingrédients, la durée de préparation, de cuisson, la recette elle-même, éventuellement son coût, le nombre de calories fournies...
Pour utiliser votre fichier, il vous faudra en général deux programmes qui rempliront deux fonctions différentes : la première consiste à enregistrer le fichier sur la disquette, c'est-à-dire le transférer de la mémoire centrale sur la disquette : cette l'*écriture*.
La deuxième opération, inverse, consiste à transférer le fichier de la disquette dans la mémoire centrale et à l'afficher à l'écran (ou sur imprimante) : c'est la *lecture*.
Ces deux fonctions pourront d'ailleurs ultérieurement être réunies dans le même programme.

Attention : lire et écrire ont toujours pour sujet implicite l'"ordinateur". Il faut toujours le sous-entendre et ne pas rapporter ces opérations de lecture et d'écriture à vous-même.

Écrire un enregistrement

Nous supposerons dans la suite qu'un seul lecteur de disquettes est utilisé. Si ce n'est pas le cas, tenez compte de ce qui a été écrit précédemment (chapitre II).
Prenons un fichier de numéros de téléphone et réduisons-le à l'essentiel : la liste des noms et des numéros correspondants.
Nous utiliserons un enregistrement par personne formé des deux chaînes de caractères *nom* et *numéro* (le numéro n'est pas un nombre : d'abord il comporte des blancs, parfois des parenthèses, et sutout il ne peut pas servir à effectuer des calculs !).
Donnons d'abord un nom au fichier : "ANNUAIRE".
Choisissons ensuite un numéro de "canal" pour transférer le fichier sur la disquette. En effet les données que vous entrez en mémoire centrale passent d'abord par une sorte de sas ou "zone tampon" (en anglais *buffer*) avant d'être transférées sur la disquette.

Écriture d'un enregistrement



C'est comme pour sortir d'une tour de vingt étages : sur le palier vous avez le choix entre plusieurs ascenseurs parallèles pour vous transférer au rez-de-chaussée.
Prenons le canal n° 1.
Voici le programme qui écrit le premier enregistrement (JULES 000 00 01) du fichier "ANNUAIRE", en passant par le canal n° 1:

```
10 ' EC-ANN          (ECRITURE ANNUAIRE)
20 OPEN "O",#1,"ANNUAIRE"
30 WRITE #1,"JULES","000 00 01"
40 CLOSE #1
```

La ligne 20 ouvre (OPEN = ouvrir) la porte de l'ascenseur n° 1 (#1) qui sort ("O" pour "OUTPUT" = sortir) de la mémoire centrale pour aller sur la disquette. Attention à bien mettre la lettre "O" et non pas le chiffre "0".
La ligne 30 écrit (WRITE = écrire) le premier enregistrement (c'est-à-dire "JULES 000 00 01") dans la zone tampon n° 1.
La ligne 40 ferme (CLOSE = fermer) la communication entre la mémoire centrale et le canal n° 1 et envoie sur la disquette le contenu de la zone tampon n° 1. Plus brièvement, on dit que le fichier est alors fermé.
Exécutez ce programme d'écriture :

RUN

Le lecteur de disquettes (il est supposé seul) se met en marche, lampe verte allumée. Rien ne se passe à l'écran, si ce n'est le OK habituel. Pour constater qu'il s'est bien écrit quelque chose sur la disquette, demandez le catalogue :

DIR

Après les programmes que vous avez déjà enregistrés, vous devez voir :

ANNUAIRE.DAT D À 1

Le fichier est bien écrit. Au nom que nous lui avons donné "ANNUAIRE", a été rajouté le suffixe ".DAT" (DATA = données). Il en sera toujours ainsi pour les fichiers de données, sauf si vous donnez vous-même un autre suffixe. Comme pour les programmes, le nom de fichier comporte au maximum 8 caractères sans virgule ni point.
La lettre "D" précise le type de fichiers : Données.
La lettre "A" précise que les données du fichier sont enregistrées en code ASCII.
Le chiffre "1" indique le nombre de "blocs" occupés par le fichier.
Vous pouvez maintenant enregistrer le programme d'écriture :

SAVE "EC—ANN"

Vérifiez l'opération avec DIR.

Lire l'enregistrement qu'on vient d'écrire

Pour lire le fichier (réduit ici à un seul enregistrement), il faut :
1. ouvrir la communication entre la mémoire centrale et la disquette par l'intermédiaire d'un canal.
2. transférer en mémoire centrale le contenu du fichier et éventuellement l'afficher à l'écran.

3. fermer la communication entre la mémoire centrale et la disquette ce qui donne :

NEW

```
10  ' LE-ANN          (LECTURE ANNUAIRE)
20  OPEN "I",#1,"ANNUAIRE"
30  INPUT #1,NOM$,TEL$
40  PRINT NOM$,TEL$
100 CLOSE #1
```

En ligne 20 l'ouverture se fait dans le sens de l'entrée ("I" = INPUT) en mémoire centrale. Nous avons pris le même canal n° 1.
En ligne 30 s'effectue le transfert des données dans deux variables NOM$ et TEL$ en mémoire centrale. La fonction INPUT# joue le même rôle que la fonction INPUT, mais à partir de la disquette ou plus généralement d'un canal, au lieu du clavier.
La ligne 40 vous permet de "voir" le contenu de ces 2 variables. La ligne 100 enfin ferme la communication avec le canal #1.
Exécutez ce programme de lecture :

RUN

Les 2 variables s'affichent à l'écran (avec le premier caractère en colonnes 0 et 13 à cause de la virgule dans l'instruction PRINT).

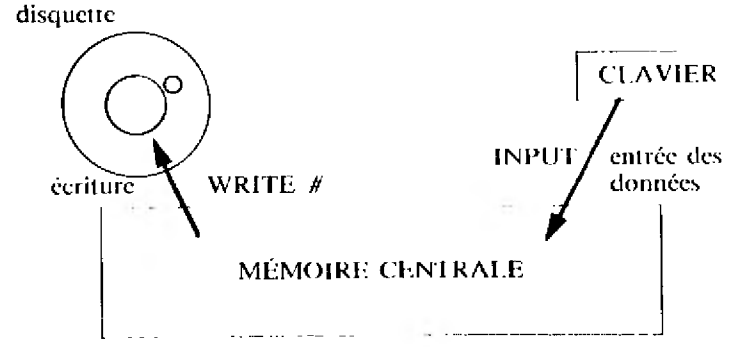
SAVE "LE—ANN"

Et vérifiez l'opération avec :

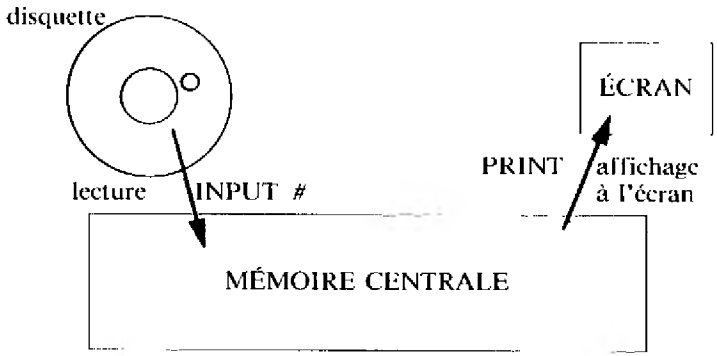
DIR

Le schéma qui suit illustre le trajet des données au cours de l'écriture et de la lecture d'un fichier par un programme en mémoire centrale.

Entrée des données et écriture



Lecture et affichage des données



Notez bien que c'est le *programme en mémoire centrale* qui "lit" ou qui "écrit" le fichier et non pas *vous*.

Écrire et lire plusieurs enregistrements

Pour inscrire un deuxième nom dans le fichier, chargeons le programme d'écriture en mémoire et rajoutons une ligne d'écriture :

```
LOAD "EC—ANN"
35 WRITE #1,"JIM","000 00 02"
```

Chaque instruction WRITE # définit un nouvel enregistrement. En faisant tourner ce programme modifié, on écrit la nouvelle version du fichier "ANNUAIRE".

```
RUN
SAVE "EC—ANN"
```

Cette nouvelle version a effacé l'ancienne : si vous demandez le catalogue, il ne comporte qu'un seul fichier "ANNUAIRE", le dernier écrit. Pour lire le fichier, il faut modifier en conséquence le programme de lecture :

```
LOAD "LE—ANN"
```

Ajoutons les 2 lignes qui correspondent à l'entrée et l'affichage d'un nouvel enregistrement :

```
50  INPUT #1,NOM$,TEL$
60  PRINT NOM$,TEL$
RUN
```


La nouvelle version du fichier s'affiche à l'écran :

```
JULES      000 00 01
JIM        000 00 02
```

Si vous voulez éviter d'avoir à enregistrer et charger en alternance le programme d'écriture et le programme de lecture (qui jouent pour un fichier de données le même rôle que les instructions SAVE et LOAD pour un fichier programme), vous pouvez les regrouper dans un seul programme.
Il est possible de fusionner les deux programmes grâce à l'instruction MERGE :
— renumérotons le programme "LE—ANN" à partir de la ligne 100 afin qu'il n'y ait pas de lignes communes aux deux programmes

```
RENUM 100
```

puis enregistrons-le sous forme ASCII :

```
SAVE "LE—ANN",A
```

— chargeons le programme "EC—ANN" en mémoire centrale :

```
LOAD "EC—ANN"
```

— et enfin fusionnons les deux programmes :

```
MERGE "LE—ANN"
LIST
```

Pour pouvoir exécuter séparément le programme d'écriture, il faut ajouter une instruction END en ligne 50.

```
10  ' EC-ANN          'ECRITURE ANNUAIRE '
20  OPEN "O",#1,"ANNUAIRE"
30  WRITE #1,"JULES","000 00 01"
35  WRITE #1,"JIM","000 00 02"
40  CLOSE #1
50  END
100 ' LE-ANN          'LECTURE ANNUAIRE '
110 OPEN "I",#1,"ANNUAIRE"
120 INPUT #1,NOM$,TEL$
130 PRINT NOM$,TEL$
140 INPUT #1,NOM$,TEL$
150 PRINT NOM$,TEL$
160 CLOSE #1
```

Pour exécuter "EC—ANN", il suffit de taper :

```
RUN
```

Et pour exécuter "LE—ANN", il faut taper :

RUN 100

Vous pouvez sauvegarder ce programme de gestion en l'appelant EL—ANN :

```
5' EL—ANN
SAVE "EL—ANN"
```

Un nombre quelconque d'enregistrements

Il est évident qu'avec le procédé précédent (une ligne pour chaque instruction WRITE en écriture et deux lignes pour lire chaque enregistrement), il serait fastidieux de rentrer un fichier important.
Pour répéter la même opération nous ferons donc une itération, c'est-à-dire une boucle, ou plutôt deux boucles : une "en écriture" et une "en lecture".
Voyons d'abord l'écriture :

NEW

```
5   ' EL-ANN
7   CLS
10  ' EC-ANN          (ECRITURE ANNUAIRE)
15  PRINT "ENTREE DES NOMS DANS LE FICHI
ER"
20  OPEN "O",#1,"ANNUAIRE"
30  INPUT "NOM:";NOM$
40  IF NOM$="" THEN 80
50  INPUT "TELEPHONE:";TEL$
60  WRITE #1,NOM$,TEL$
70  GOTO 30
80  CLOSE #1
90  PRINT "ECRITURE TERMINEE"
95  END
```

Les lignes 30 à 70 constituent une boucle d'écriture des enregistrements. La boucle est terminée lorsqu'on entre un nom vide, c'est-à-dire lorsqu'on appuie directement sur ENTRÉE en réponse à la question posée ligne 30.
Essayez le programme :

RUN

Et répondez avec une liste de noms et numéros de téléphone. Voici maintenant le programme de lecture avec sa boucle :

```
100 ' LE-ANN          (LECTURE ANNUAIRE)
110 OPEN "I",#1,"ANNUAIRE"
120 INPUT #1,NOM$,TEL$
130 PRINT NOM$,TEL$
140 GOTO 120
150 CLOSE #1
```

Faites le tourner :

RUN 100

S'affichent à l'écran les noms et numéros que vous venez d'écrire. Mais à la fin vous obtenez le message d'erreur :

?IE Error in 120 ou Error 54 in 120

pour "Input past End" = lecture après la fin du fichier.
En effet, il n'y a pas de test pour arrêter la boucle : lorsque le dernier enregistrement a été transféré en mémoire centrale, l'instruction INPUT # ne trouve plus rien, d'où le message d'erreur.
Une fonction permet de savoir si un fichier est terminé : c'est la fonction EOF (End Of File = fin de fichier).
Tant que le fichier qui transite par le canal n° 1 n'est pas terminé, EOF(1) vaut 0 soit l'état "FAUX". Lorsqu'on atteint la fin du fichier, alors EOF(1) vaut —1, soit l'état "VRAI".
Le test s'écrit donc :

```
115 IF EOF(1) THEN 150
140 GOTO 115
```

RUN 100

Cette fois la lecture du fichier se termine sans message d'erreur.

Modifier un enregistrement

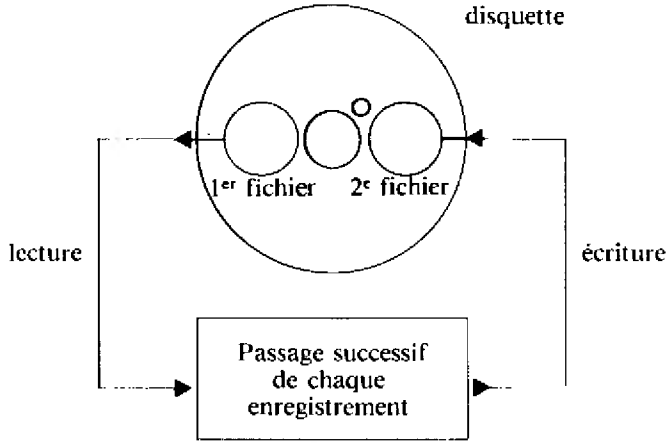
D'après ce que nous venons de voir, avec un fichier séquentiel vous êtes ou bien en *écriture* ou bien en *lecture* : il est impossible d'être à la fois en "sortie" ("O") et en "entrée" ("I"). La communication entre la disquette et la mémoire centrale est à sens unique (il n'en sera pas de même avec les fichiers à accès direct).

Comment faire alors pour modifier une donnée au milieu du fichier ? (Vos amis déménagent, les PTT changent les numéros...).
La seule solution est de créer un deuxième fichier qui sera identique au premier sauf pour les enregistrements à modifier. Le fichier original sera transféré en mémoire centrale à travers un canal (n° 1 par exemple) et recopié sur le deuxième fichier à travers un autre canal (n° 2 par exemple). Seules les données à modifier ne seront pas recopiées du premier sur le deuxième fichier : chaque numéro de téléphone modifié sera entré en mémoire centrale par le clavier puis écrit dans le deuxième fichier.

Voici la structure du programme de modification :
— ouvrir le fichier "ANNUAIRE" en lecture sur le canal n° 1
— ouvrir un nouveau fichier "FICNOUV" en écriture sur le canal n° 2

pour tous les enregistrements :
- lire l'enregistrement NOM$, TEL$ (si le fichier 1 n'est pas terminé) et afficher son contenu à l'écran
- demander si cet enregistrement doit être modifié, si oui entrer la modification
- écrire l'enregistrement sur le nouveau fichier

— fermer le canal n° 2 en écriture
— fermer le canal n° 1 en lecture
ou suivant le schéma suivant :



Il est évident que les ronds symbolisant chaque fichier sur la disquette ne représentent pas la surface réelle de mémoire occupée !

```
10  ' MOD-ANN1 (MODIFICATION ANNUAIRE)
20  OPEN "I",#1,"ANNUAIRE"
30  OPEN "O",#2,"FICNOUV"
40    IF EOF(1) THEN 120
50    INPUT #1,NOM$,TEL$
60    PRINT NOM$,TEL$
70    INPUT "MODIFICATION (O/N)";REP$
80    IF REP$="N" THEN 100
90    INPUT "NOUVEAU NUMERO:";TEL$
100   WRITE #2,NOM$,TEL$
110   GOTO 40
120 CLOSE #1,#2
```

Remarquez en ligne 120 la fermeture des deux canaux en un seul coup. Vous pouvez même écrire :

120 CLOSE

ce qui ferme tous les fichiers ouverts.
Essayez le programme : vous pouvez modifier tous les numéros de téléphone que vous voulez.
Lorsque vous avez terminé de parcourir votre annuaire, vous avez deux fichiers, l'ancien et le nouveau (air connu...). Vérifiez-le sur le catalogue de la disquette. Il suffit d'éliminer l'ancien et de rebaptiser ANNUAIRE la nouvelle version :

KILL "ANNUAIRE.DAT"
NAME "FICNOUV.DAT" AS "ANNUAIRE.DAT"
(Dans les instructions KILL et NAME, les suffixes sont indispensables.)

Vérifiez le résultat de l'opération sur le catalogue.
Et enfin sauvegardez votre programme de modification :

SAVE "MOD—ANN1

Si vous n'avez qu'un seul numéro à modifier, il est un peu long de parcourir tout l'annuaire. Vous pouvez modifier le programme afin qu'il demande le nom correspondant au numéro à modifier :

```
10  ' MOD-ANN2 (MODIFICATION ANNUAIRE)
14  INPUT"NOM A MODIFIER";N$
16  INPUT"NOUVEAU NUMERO";T$
60  IF NOM$=N$ THEN TEL$=T$ ELSE 100
70  PRINT "MODIFICATION FAITE "
```

et supprimez les lignes 80 et 90

LIST
RUN

Si le message "MODIFICATION FAITE" n'apparaît pas, c'est que le nom N$ n'est pas dans l'annuaire. Sauvegardez cette deuxième version de votre programme de modification :

SAVE "MOD—ANN2"

En vous souvenant qu'il n'est pas possible à la fois de lire et écrire dans un fichier séquentiel, vous comprenez comment on peut modifier un fichier séquentiel : en le réécrivant. Pour supprimer une partie, on recopie tout sauf cette partie.
Pour ajouter de nouveaux noms, on recopie tout et on ajoute à la fin les noms avec leurs numéros de téléphone.

Remarque : n'arrêtez pas le programme de modification au milieu de l'exécution ; votre annuaire ne serait qu'à moitié recopié. Si tel était le cas, il vous faudrait tout reprendre depuis le début avant de faire KILL puis NAME.

Un fichier de nombres : les points d'un dessin

Les coordonnées des points d'un dessin peuvent être placées dans une instruction DATA. Mais en les mettant dans un fichier, on peut les appeler dans n'importe quel programme, et utiliser ce dessin quand on veut sans avoir à réécrire la liste des coordonnées.
Jusqu'à présent les données étaient des chaînes de caractères. Le procédé est le même pour écrire un fichier de nombres :

```
10 ' EC-DESS
20 OPEN "O",#1,"DESSIN"
30 INPUT "COL,LIG";COL,LIG
40 IF COL=-1 THEN 70
50 WRITE #1,COL,LIG
60 GOTO 30
70 CLOSE #1
```

RUN

Lorsque les coordonnées du dernier point ont été écrites, donnez à COL la valeur —1 (et n'importe quelle valeur à LIG) pour arrêter l'écriture du fichier.

SAVE "EC—DESS"

A la lecture, le premier point fixé par PSET (C,L) est celui où doit commencer le dessin :

```
10 ' LE-DESS
15 CLS
20 OPEN "I",#1,"DESSIN"
30 INPUT #1,C,L
40 PSET (C,L)
50 IF EOF(1)=-1 THEN 90
60 INPUT #1,COL,LIG
70 LINE -(COL,LIG)
80 GOTO 50
90 CLOSE
```

RUN

Le programme de lecture n'affiche pas les valeurs des variables mais les utilise directement, ici dans une instruction LINE.

SAVE "LE-DESS"

Les fichiers de nombres s'écrivent et se lisent exactement comme les fichiers de chaînes de caractères : avec WRITE# pour l'écriture et INPUT# pour la lecture.
Vous pouvez dont mélanger des données chaînes de caractères et des données numériques : il suffit que les variables qui suivent une instruction INPUT# correspondent bien aux variables qui suivent l'instruction WRITE# correspondante.
Par exemple :

```
WRITE# 1, NOM$, PRENOM$, AGE          (en écriture)
INPUT # 1, NOM$, PRENOM$, AGE         (en lecture)
```

Utiliser plusieurs fichiers de données de même nature

Si vous avez 10 dessins différents stockés chacun dans un fichier, il vous faudra à priori 10 programmes d'écriture et 10 programmes de lecture correspondants (éventuellement encore 10 programmes de modification). Or dans ces programmes, la seule chose qui change est le nom du fichier, c'est-à-dire du dessin.
Le bon réflexe en programmation, est de toujours placer dans des variables "ce qui change".

Modifions donc "EC—DESS" pour en faire l'unique programme destiné à l'écriture de tous les dessins :

```
10 ' EC-DESS
15 INPUT "NOM DU DESSIN ";DESSIN$
20 OPEN "O",#1,DESSIN$
```

et de même pour le programme de lecture :

```
10 ' LE-DESS
15 INPUT "NOM DU DESSIN ";DESSIN$
20 OPEN "I",#1,DESSIN$
```

De l'importance de la fermeture des fichiers (UNLOAD)

Au cours de l'écriture de fichiers vous avez peut-être constaté que le lecteur ne tourne pas lors de l'entrée de chaque enregistrement. Si vous écrivez peu de données, le moteur démarre seulement après l'entrée de la dernière donnée. Si vous écrivez un grand nombre de données, le moteur démarre de temps en temps. (Remplissez un fichier de 200 ou 300 zéros pour le vérifier !)
En effet, l'ordinateur écrit les données dans une zone tampon (buffer) et ne transfère ces données sur la disquette que lorsque la zone tampon est pleine (ou lorsqu'il reçoit l'ordre CLOSE).
La capacité d'une zone tampon étant d'un secteur (128 octets), le transfert des données de la zone tampon à la disquette se fait donc 128 caractères par 128 caractères.
L'instruction CLOSE a pour objet d'écrire les dernières données de la zone tampon (celle-ci n'étant pas pleine à la fin du fichier) et aussi de noter dans le catalogue de la disquette l'endroit de la disquette où s'arrête effectivement le fichier.
Cette dernière information est très importante lors de la lecture du fichier.
Si vous vous arrêtez au milieu du programme d'écriture à cause d'une erreur, ou parce que vous avez appuyé sur CNT C, la fermeture du fichier ne sera pas faite.
Si vous ôtez la disquette immédiatement après, l'ordinateur risque de ne plus s'y retrouver : les informations concernant ce fichier (en particulier l'endroit où il se termine) n'auront pas été inscrites sur le catalogue puisque l'instruction CLOSE n'a pas été exécutée. Un autre incident grave risque de se produire si vous mettez alors une autre disquette dans le lecteur. En effet, lors de la prochaine opération sur fichier, les informations restées en suspens sur la première disquette seront alors écrites sur la seconde, ce qui la brouille ainsi que son catalogue.

Il existe une commande qui vous garantit contre toute erreur de ce genre : UNLOAD (= "décharger").
Si vous tapez :

```
UNLOAD
```

avant de retirer une disquette, vous serez sûr que tous les fichiers seront correctement fermés et toutes les informations les concernant dûment enregistrées sur le catalogue.
A priori cette commande s'applique au lecteur n° 0. Pour qu'elle agisse sur le lecteur n° 1, il suffit d'écrire :

```
UNLOAD 1
```

Recensement général des périphériques utilisables avec un fichier

Vous savez maintenant écrire et lire un fichier sur disquette. D'autres périphériques peuvent être utilisés, mais il faut alors le préciser. En effet en l'absence d'autre indication, lorsque le lecteur de disquettes (n° 0) est branché, il est pris comme le périphérique implicitement souhaité. Si vous voulez utiliser un autre périphérique que le lecteur n° 0, (lecteur n° 1, cassette...)
il faut l'indiquer dans l'instruction d'ouverture du fichier :

en écriture

```
OPEN "O", #1, "1 :ANNUAIRE"
pour écrire sur le lecteur n° 1,
OPEN "O", #1, "CASS :ANNUAIRE"
pour écrire sur la cassette,
OPEN "O", #1, "LPRT :"
pour écrire sur imprimante,
OPEN "O", #1, "SCRN :"
pour écrire à l'écran,
OPEN "O", #1, "COMM :"          (sur TO 7)
pour écrire la voie de communication série,
```

L'ensemble "CASS : ANNUAIRE" constitue le *descripteur de fichier.*

Pour les trois derniers périphériques, il est inutile de préciser le nom du fichier puisqu'il est seulement transmis et non pas conservé dans une mémoire.

De même en lecture :

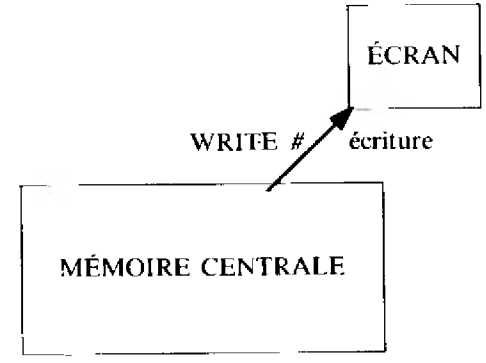
en lecture

OPEN "1", #1, "1 : ANNUAIRE"
pour lire à partir du lecteur n° 1,
OPEN "1", #1, "CASS : ANNUAIRE"
pour lire à partir de la cassette,
OPEN "1", #1, "KYBD :"
pour lire à partir du clavier (KEYBOARD)
OPEN "1", #1, "COMM :" *(sur TO 7)*
pour lire à partir de la voie série.

Il est évidemment impossible de "lire" un fichier écrit à l'écran ou sur l'imprimante : ces deux périphériques sont exclusivement des périphériques "de sortie".
Inversement, il est impossible d'"écrire" sur le clavier : ce dernier est un périphérique d'"entrée".
Seuls les lecteurs de disquettes, de cassettes, et la voie de communication série peuvent être utilisés aussi bien en entrée qu'en sortie.
Deux périphériques sont un peu différents des autres : l'écran et le clavier. En effet, les deux instructions du Basic standard INPUT et PRINT permettent à l'ordinateur de "lire" à partir du clavier (entrée des données) ou d'"écrire" à l'écran (affichage) sans utiliser de canal.
C'est pour mettre ces deux périphériques en parallèle avec les autres que vous est offerte la possibilité d'ouvrir un fichier en lecture avec le clavier (KYBD :) et en écriture avec l'écran (SCRN :).

Voir le fichier s'écrire à l'écran

Jusqu'à présent vous n'avez pas encore vu le *fichier* lui-même, c'est-à-dire ce qui est effectivement inscrit sur la disquette.
Pour voir un fichier, utilisons l'écran comme périphérique de sortie en écriture plutôt que le lecteur de disquettes. Inutile de vous dire que ceci ne présente aucun intérêt en pratique car si l'on écrit un fichier, c'est en général pour le garder. Et sur l'écran, la durée de vie du fichier est limitée au déroulement des 24 lignes !



Pour écrire le fichier "ANNUAIRE", chargez en mémoire centrale le programme d'écriture correspondant :

```
LOAD "EL—ANN"
```

Le périphérique implicite est le lecteur de disquettes (n° Ø).
Modifions donc la ligne 2Ø :

```
2Ø OPEN "O", #1, "SCRN : ANNUAIRE"
```

Et pour mieux voir encore le fichier lui-même, écrivons-le en rouge sur fond jaune (en supposant que vous utilisez les couleurs standard bleu sur cyan) :

```
55 COLOR 1,3
6Ø WRITE #1,NOM$,TEL$
65 COLOR 4,6

RUN
```

Dans la sortie sur l'écran, les éléments du fichier s'affichent en rouge sur fond jaune.
Remarquez que l'instruction WRITE écrit les chaînes de caractères entre guillemets. En outre, à la fin de chaque enregistrement elle inscrit les

caractères "retour au début de la ligne" et "saut de ligne" : le curseur revient au début de la ligne suivante après chaque enregistrement. Cependant sur la disquette, tous les enregistrements sont à la suite les uns des autres, la fin d'un enregistrement étant justement définie par ces caractères "retour à la ligne".

SAVE "EL—ANN2"

Comparez maintenant avec ce qui se passe pour l'écriture d'un fichier de nombres :

LOAD "EC—DESS"

```
20 OPEN "O",#1,"SCRN:DESSIN"
45 COLOR 3,4
55 COLOR 6,4
```

Les données numériques sont écrites dans le fichier sans guillemets et, comme les chaînes de caractères, séparées par une virgule.

Recopier un fichier sur cassette

Il peut être intéressant de conserver un fichier sur cassette, par exemple pour en garder une copie de sécurité. La gestion d'un fichier sur cassette se fait de la même manière que sur disquette : un programme sert à écrire le fichier (on est alors "en sortie") et un autre programme sert à le lire (on est alors "en entrée").
La seule différence réside dans le "descripteur de fichier" :

OPEN "O", #1, "CASS : ANNUAIRE"

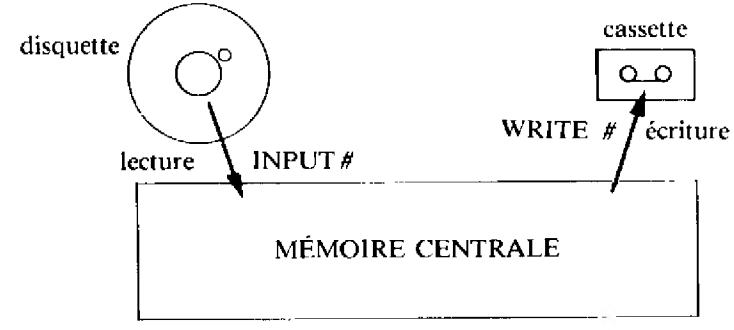
pour ouvrir le fichier en écriture,

OPEN "I", #1, "CASS : ANNUAIRE"

pour ouvrir le fichier en lecture.
Vous pouvez ainsi écrire le programme qui transfère un fichier d'une disquette (dans le lecteur n° Ø) sur une cassette. Il faut :
— ouvrir le fichier sur disquette en lecture (sur un canal)
— ouvrir le fichier sur cassette en écriture (sur un autre canal).
— pour tous les enregistrements :
• lire un enregistrement de la disquette
• écrire cet enregistrement sur la cassette
— fermer les deux canaux


```
10 ' DIS-CASS
20 OPEN "I",#1,"ANNUAIRE"
30 OPEN "O",#2,"CASS:ANNUAIRE"
40   IF EOF(1)=-1 THEN 80
50   INPUT #1,NOM$,TEL$
60   WRITE #2,NOM$,TEL$
70   GOTO 40
80 CLOSE
```

Le schéma du transfert des données est le suivant :



Pour ensuite recopier le fichier de la cassette sur la disquette, il suffit d'inverser les deux descripteurs de fichier dans le schéma et le programme précédents :

```
10 ' CASS-DIS
20 OPEN "I",#1,"CASS:ANNUAIRE"
30 OPEN "O",#2,"ANNUAIRE"
```

Et si votre lecteur de disquettes est en panne

Ou comment se débrouiller avec seulement le Basic en mémoire morte et le lecteur de cassettes ?
Si vous ne respectez pas l'ordre d'allumage et d'extinction des périphériques et du micro-ordinateur, ou pire encore, si vous débranchez un fil alors que tout est allumé, vous courez le risque d'aller au devant de problèmes sérieux.
Si vous prêtez votre lecteur de disquettes à un ami, vous risquez d'être dans la même situation : sans Basic Disque.
Mais avec le Basic en mémoire morte, vous pouvez aussi enregistrer des fichiers sur cassette.

Les instructions d'ouverture et de fermeture de fichiers sont identiques. La seule différence avec le Basic Disque est que l'instruction WRITE # n'existe pas. C'est l'instruction PRINT # qui la remplace ; cette instruction peut d'ailleurs être aussi utilisée (*avec précaution comme nous le verrons plus loin*) avec le lecteur de disquettes.
Lorsqu'aucun lecteur de disquettes n'est raccordé au micro-ordinateur, le périphérique de sortie implicite (c'est-à-dire celui qui est pris lorsqu'on ne précise rien dans le descripteur de fichier) est le lecteur de cassettes.
Les programmes de lecture et d'écriture d'un fichier (par exemple "ANNUAIRE") sont donc identiques sauf pour l'instruction d'écriture.
Sauvegardez sur cassette le programme "EL—ANN" et changez WRITE # en PRINT # dans la ligne 60 :

60 PRINT #1, NOM$, TEL$

La lecture se fait de la même manière par INPUT # et PRINT.
Enregistrez ce programme de gestion de fichiers sur cassette puis avancez la bande et placez le lecteur en "enregistrement" pour "écrire" le fichier (en notant le n° du compteur).

RUN

Après avoir écrit le nom du fichier sur la bande (il n'y a pas de catalogue sur cassette), l'ordinateur attend vos données qu'il va écrire les unes après les autres.
Lorsque l'écriture du fichier est terminée, vous pouvez le relire : replacez la bande un peu avant le numéro correspondant au début du fichier et tapez :

RUN 100

Les données s'affichent à l'écran.

Gagner de la place en mémoire grâce à PRINT

Si vous voulez utiliser au maximum la mémoire de la disquette, vous allez pouvoir réduire un peu la place occupée par les données numériques en utilisant PRINT # au lieu de WRITE #.
En effet PRINT # écrit de la même manière que PRINT affiche à l'écran :
— sans guillemets autour des chaînes de caractères
— avec les mêmes règles pour les "séparateurs" de données : point virgule (données les unes à la suite des autres) et virgule (données présentées toutes les 13 colonnes).

L'inconvénient principal de PRINT # est de ne pas mettre les chaînes entre guillements (c'est d'ailleurs pour cela que WRITE # a été inventé !). Cependant c'est grâce à cette absence de guillemets que nous allons gagner de la place en mémoire.
Voyons ce qu'on peut faire avec un fichier de nombres ("DESSIN") :

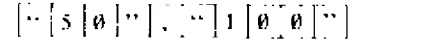
LOAD "EC—DESS

et modifions ce programme pour utiliser l'écran comme périphérique d'écriture, en y introduisant les instructions suivantes :

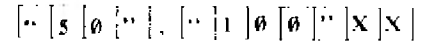
```
10 ' EC-DESS2
20 OPEN "O",#1,"SCRN:DESSIN"
45 COLOR 1,3
50 WRITE #1,COL;LIG
```

RUN

Le point de coordonnées (50, 100) est écrit ainsi à l'écran :



quand le fichier est écrit sur disquette, l'enregistrement est :

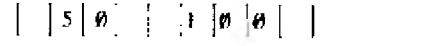


en représentant par deux croix les caractères "retour au début de la ligne" et "saut à la ligne suivante". Une suite d'enregistrements sur disquette est donc :

Utilisons maintenant PRINT # en utilisant le point-virgule comme séparateur des données :

50 PRINT # 1,COL;LIG

Un enregistrement à l'écran devient :



En effet, le point-virgule entre les variables commande leur affichage les unes à la suite des autres.
A la lecture, l'instruction INPUT # reconnaît la présence de deux nombres s'ils sont séparés par (au moins) un blanc. Vous pouvez le vérifier en écrivant le fichier sur disquette (supprimez "SCRN :") puis en le relisant.
Pour gagner de la place supprimons l'espace qui suit chaque nombre en écrivant STR$ (COL) et STR$ (LIG) au lieu de COL et LIG ; STR$ (COL) est la chaîne de caractères formée du nombre précédé d'un espace.

```
42 COL$=STR$(COL)
44 LIG$=STR$(LIG)
45 COLOR 1,3
50 PRINT #1,COL$;LIG$
55 COLOR 4,6
```

L'enregistrement devient :

	5	0		1	0	0

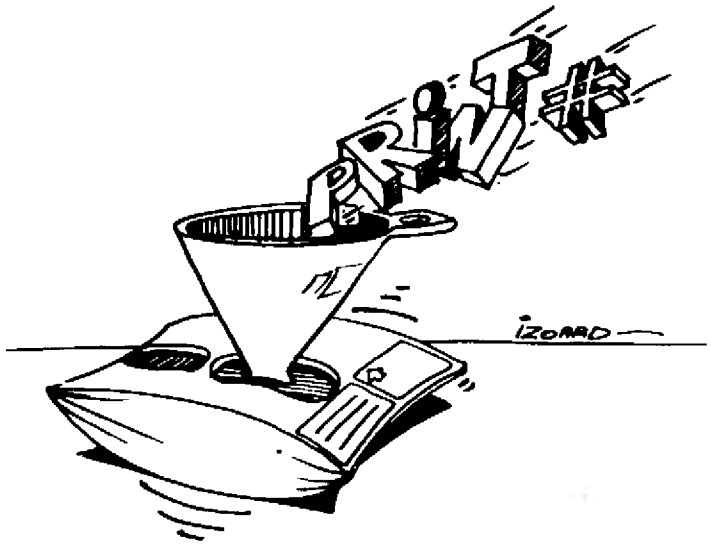
Encore mieux ! supprimons l'espace devant le numéro de colonne avec la fonction MID$;

```
42 COL$=MID$(STR$(COL),2)
```

Cette fois on ne peut plus rien supprimer !

5	0		1	0	0

Notre enregistrement occupe 8 octets (6 + 2) au lieu de 12 (10 + 2) avec WRITE #



Ainsi l'instruction PRINT # permet de gagner de la place sur la disquette. Mais prenez garde lorsque vous utilisez PRINT # avec un point-virgule ou même une virgule comme séparateur : avec un point virgule, si vous avez des données *chaînes de caractères*, elles seront enchaînées les unes aux autres sans espace ni rien pour les séparer. Il faudra alors que vous

rajoutiez une virgule entre les données afin qu'à la lecture, elles soient bien prises une par une.

Par exemple :

```
PRINT # 1,NOM$;",";TEL$
```

En utilisant PRINT # avec une virgule comme séparateur, les chaînes de caractères ne seront séparées que par un caractère.

Utilisez PRINT # USING pour bien présenter vos données sur imprimante

Si vous ne savez pas où s'envole votre argent et si vous avez la volonté et la persévérance nécessaires pour tenir le compte de vos dépenses, vous pouvez noter ces dernières dans un fichier. Vous pourrez ensuite les regrouper, faire des moyennes...
Élaborons d'abord le programme d'écriture sur disquette de ce fichier de dépenses :

```
10 ' EC-DEP
20 OPEN "O",#1,"DEPENSES"
30 WRITE #1,160,50
40 WRITE #1,78,80
50 CLOSE #1
```

Attention, si vous écrivez 160,50 (avec une virgule au lieu d'un point), l'ordinateur écrira deux nombres différents 160 et 50.

```
RUN
```

Ça y est, le fichier est écrit.

```
SAVE "EC-DEP"
```

Pour afficher les données à l'écran, il suffit de relire les données (INPUT #) et de les afficher (PRINT) :

```
10 ' LE-DEP
20 OPEN "I",#1,"DEPENSES"
30   IF EOF(1) THEN 70
40   INPUT #1,PRIX
50   PRINT PRIX
60   GOTO 30
70 CLOSE #1
```

```
RUN
```

Mais peut-être préfériez-vous que les données s'affichent de manière plus lisible :

```
150.50 F
 78.80 F
```

Pour cela, ajoutez la ligne 25 et modifiez la ligne 50

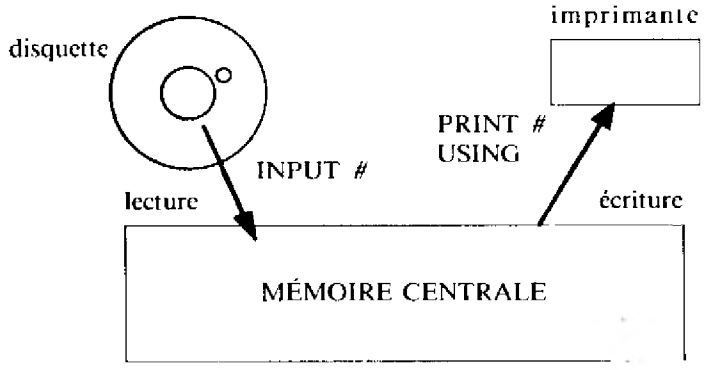
```
25 PRE$="####.## F"

50   PRINT USING PRE$;PRIX
```

L'instruction PRINT USING permet d'afficher des données suivant une disposition (ou un "format") prévue à l'avance.
Essayez avec un nombre :

```
PRINT USING "# #.# #";5
```

Le nombre est affiché suivant le modèle du format : deux chiffres avant le point décimal et deux chiffres après.
Essayez avec d'autres nombres.
Pour notre fichier, nous avons même ajouté la lettre "F" à la fin de chaque prix : on peut ajouter des caractères dans le format.
Si vous voulez garder une trace écrite de votre fichier, il va falloir écrire le fichier sur imprimante et donc ouvrir un fichier en écriture sur l'imprimante.

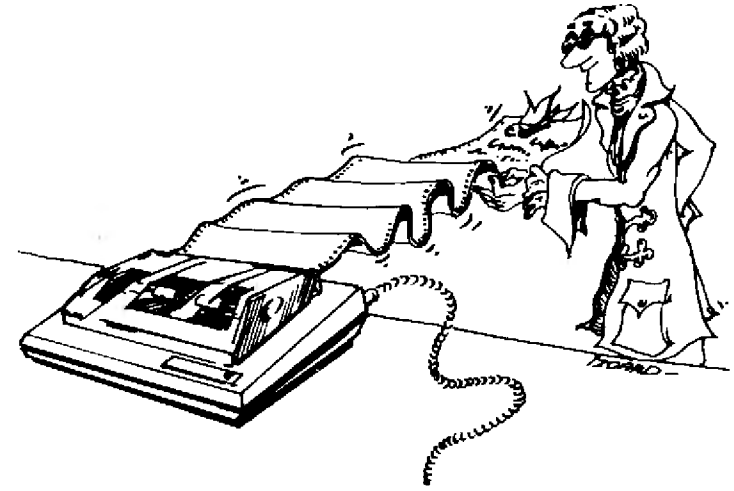



```
10 ' LE-DEP
20 OPEN "I",#1,"DEPENSES"
22 OPEN "O",#2,"LPRT "
25 PRE$="####.## F"
30   IF EOF(1) THEN 70
40   INPUT #1,PRIX
50   PRINT #2,USING PRE$;PRIX
60   GOTO 30
70 CLOSE #1,#2
```


Résultat : une belle impression sur papier à garder dans vos classeurs !
Si l'imprimante n'est pas en marche, le programme se bloque.
Appuyez sur la touche d'initialisation pour retrouver votre programme et sauvegardez le :

```
SAVE "LE-DEP"
```

Mais maintenant il vous faut allumer l'imprimante, alors attention à l'ordre d'allumage.
— retirez la disquette du lecteur,
— éteignez l'unité centrale puis les autres périphériques,
— branchez l'imprimante et allumez-la,
— rallumez les autres périphériques,
— rallumez l'unité centrale,
— remettez la disquette dans le lecteur.
Pour éviter toute cette gymnastique, allumez tous les périphériques que vous risquez d'utiliser dès le début de votre travail.



Règle de bon usage

Si vous arrêtez un programme d'écriture de fichier par CNT C, ou s'il s'arrête par un message d'erreur, fermez tous les fichiers en mode direct, et recopiez toutes les interventions récentes sur la disquette par :

```
UNLOAD
```

Vous pourrez alors retirer la disquette du lecteur sans perdre son contenu.

Chapitre V / Les fichiers à accès direct

Les fichiers à accès direct ont été créés pour tirer un meilleur parti de la disquette (ou des disques sur des ordinateurs un peu plus gros). En effet, dans un fichier séquentiel, pour lire le 100e enregistrement, il faut commencer au début et lire d'abord les 99 autres avant d'y arriver. Dans un fichier à accès direct, on va directement au 100e enregistrement si l'on veut. C'est un peu comme un cahier pour noter des noms. Si vous écrivez tous les noms sur le même cahier les uns à la suite des autres, il vous faut relire tout depuis le début pour arriver à celui que vous cherchez. Si vous notez vos noms dans un répertoire, il suffit par la suite d'aller à la lettre qui vous intéresse pour les retrouver.
L'inconvénient du fichier à accès direct, c'est que chaque case du fichier doit être parfaitement définie à l'avance (comme les découpages dans le répertoire) et en outre chaque enregistrement doit avoir la même longueur.

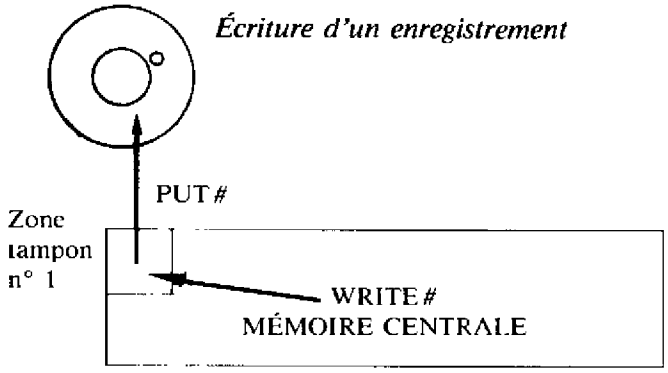
Une chose pour chaque jour, un jour pour chaque chose : un agenda simplifié

Assez de généralités. Prenons un exemple concret. Nous allons essayer de transformer notre micro-ordinateur en agenda. Pas question de le transporter dans la poche intérieure de votre blouson ! Nous nous contenterons dans un premier temps d'un agenda de bureau.
La fonction principale d'un agenda est de pouvoir noter, quand ça se présente, tout ce à quoi il faut penser les jours suivants. Pour simplifier, nous allons nous limiter à un mois entier, c'est-à-dire 31 jours, et nous allons noter un seul événement par jour. En prenant pour nom du fichier le nom du mois en question ("mai" par exemple), comment faire pour écrire que le 9, il ne faut pas oublier le cours d'anglais ?
La solution est très proche d'un fichier séquentiel.

```
10   AGENDA ECRITURE
20  OPEN "D",#1,"MAI"
30  WRITE #1,"ANGLAIS"
40  PUT #1,9
100 CLOSE #1
RUN
```

Comme pour tout fichier, l'écriture d'un enregistrement comporte trois phases : l'ouverture du fichier (OPEN), l'écriture des données et la fermeture du fichier (CLOSE). La première différence se voit à l'ouverture : OPEN est suivi de "D" (pour Direct) ou encore "R" (pour Random).
La deuxième différence réside dans l'écriture qui se fait en deux temps. D'abord WRITE# écrit la donnée dans la zone tampon réservée au fichier, puis PUT# transfère le contenu de la zone tampon (qui constitue un enregistrement) sur la disquette. Tous les enregistrements auront donc la même longueur, celle de la zone tampon soit 128 octets. L'instruction PUT# indique dans quel fichier il faut écrire (ici le canal n° 1) et dans quel enregistrement (ici le neuvième).

Écriture d'un enregistrement



Pourquoi cette deuxième instruction pour écrire ?
Dans un fichier séquentiel, après WRITE#, le BASIC sait où il doit écrire les données dans le fichier : à la suite des précédentes (ou au début pour les premières). Dans un fichier à accès direct, il faut une instruction pour indiquer où les écrire, puisqu'on a le droit de les écrire n'importe où.
Essayons d'en écrire un peu plus. Le 14, vous avez prévu une randonnée à bicyclette ?

```
50  WRITE #1,"RANDO A VELO"
60  PUT #1,14
```

Faites le tourner :

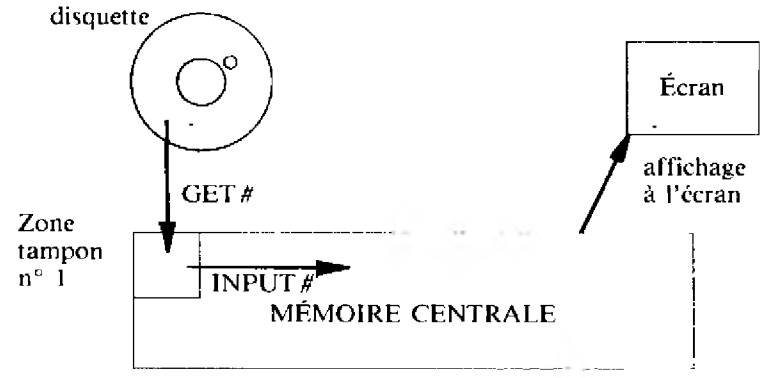
RUN

Il est temps maintenant de relire ce qui a été écrit.
En voici le programme :

```
10  ' AGENDA LECTURE
20  OPEN "D",#1,"MAI"
30  GET #1,9
40  INPUT #1,QUOI$
50  PRINT "LE 9 ";QUOI$
60  GET #1,14
70  INPUT #1,QUOI$
80  PRINT "LE 14 ";QUOI$
100 CLOSE #1
```

L'ouverture est identique en lecture et en écriture :
OPEN "D", ..., ce qui veut dire qu'un fichier à accès direct est utilisable, *sur le même canal*, en lecture et en écriture ; nous y reviendrons.
La lecture des données se fait également en deux temps. D'abord le transfert par GET# de l'enregistrement n° 9 dans la zone tampon n° 1. Ensuite la lecture des données situées dans cette zone tampon par INPUT# comme pour un fichier séquentiel.

Lecture d'un enregistrement



Il faut remarquer tout de suite la correspondance entre PUT# et GET# d'une part et entre WRITE# et INPUT# d'autre part. D'ailleurs, pour le même enregistrement, les données lues par INPUT# doivent correspondre aux données écrites avec WRITE#. Dans cet exemple, il n'y a qu'une donnée, c'est facile. Mais si vous voulez écrire des nombres et des chaînes de caractères avec WRITE# dans l'enregistrement, il faudra les relire dans le même ordre avec INPUT# (problème que nous avons déjà rencontré avec les fichiers séquentiels).

Il serait intéressant de pouvoir consulter notre fichier agenda pour n'importe quel jour du mois. Reprenons pour cela notre programme de lecture. Comment le modifier pour qu'il vous demande "QUEL JOUR ?" et affiche à l'écran le contenu de l'enregistrement correspondant à ce jour ?

```
30   INPUT"QUEL JOUR";J
50   GET #1,J
60   INPUT #1,QUOI$
70   PRINT J,QUOI$
```

et pour consulter plusieurs jours ajoutez ces deux lignes :

```
40 IF  J=0 THEN 100
80 GOTO 30
```

Essayez ce nouveau programme. Tant que vous n'indiquez pas un nombre nul, vous obtenez le contenu de l'agenda ou plutôt, le contenu de ce qui a été écrit pour les jours 9 et 14 et des choses bizarres pour les autres. Prenez par exemple le jour 3 ; résultat : environ trois lignes de caractères quelconques.
En demandant ainsi la lecture d'enregistrements qui n'ont pas été écrits, vous pouvez obtenir dès effets curieux : la touche [RAZ] n'efface plus la totalité de l'écran, le défilement des lignes à l'écran ne se fait plus normalement... Une seule solution : appuyez sur la touche [INITIALISATION], et reprenez votre programme.
Pour le jour 25, vous obtenez IE error in 50 (Input past End) ou Error 54 in 50.
Apparemment, pour 25, nous avons essayé de lire au-delà de la fin du fichier.
Il existe une fonction qui va nous éviter de lire au-delà du dernier enregistrement connu. LOF(1) donne le numéro du dernier enregistrement du fichier numéro 1 (Last Of File = dernier du fichier). Vous pouvez ajouter :

```
35 IF J > LOF(1) THEN 30
```

```
10   ' AGENDA LECTURE
20   OPEN "D",#1,"MAI"
30       INPUT"QUEL JOUR";J
35       IF J> LOF(1) THEN 30
40       IF J=0 THEN 100
50       GET #1,J
60       INPUT #1,QUOI$
70       PRINT J,QUOI$
80       GOTO 30
100  CLOSE #1
```


Ça ira déjà mieux. Mais cette modification ne nous satisfait pas totalement. La meilleure solution va être d'initialiser avant usage le fichier agenda.
Sauvegardez le programme de lecture et écrivez ce programme d'initialisation du fichier. Il va remplir à l'avance tous les enregistrements du fichier par des espaces pour ne pas récupérer de choses bizarres quand il n'y a rien dans l'agenda pour un jour précis.

Si nous sommes toujours à préparer l'agenda du mois de mai, il y a 31 jours donc 31 enregistrements. Nous allons donc remplir chaque enregistrement par des espaces. Rappelons qu'à priori, la longueur d'un enregistrement est de 128 octets. Vous savez tout ce qu'il faut pour écrire le programmme. Allez-y.
Voici notre propre solution :

```
10    INI-AGEN
20  OPEN "D",#1,"MAI"
30  FOR J=1 TO 31
40    WRITE #1,SPACE$(128)
50    PUT#1,J
60  NEXT J
70  CLOSE #1
```

L'exécution est à peine lancée que nous obtenons le message :
ER error in 40 (End of Record)
Eh oui ! Nous avions oublié que l'instruction WRITE# écrit les chaînes de caractères entre guillemets et ajoute à la fin les deux caractères : "saut de ligne" et "retour en début de ligne". Vouloir écrire 132 caractères dans un enregistrement de 128 nous a déclenché une erreur. Il suffit donc de corriger la ligne 40 :

```
40    WRITE #1,SPACE$(124)
```

Cette fois, vous pouvez reprendre le programme de lecture. Immédiatement après l'initialisation, pour chaque jour demandé, vous obtenez un peu plus de trois lignes d'espaces. C'est correct.
L'initialisation d'un fichier à accès direct n'est pas obligatoire. Mais si vous êtes amené à consulter des enregistrements dans lesquels vous n'avez pas écrit, elle est bien préférable. De plus, comme tous les enregistrements sont initialisés avec la même valeur, vous pouvez facilement déterminer, par programme, si l'un d'eux a reçu des informations.
Puisqu'un agenda sert autant à écrire des notes qu'à les relire, regroupons nos deux programmes de lecture et d'écriture en un seul.

Ce programme général AGENDA aura la structure suivante :
1. Ouvrir le fichier du mois.
2. Choisir entre Lecture, Écriture ou Rien.
3. Si Rien, fermer le fichier et fin du programme.
4. Si Lecture, demander le jour et donner le contenu de l'enregistrement.
5. Si Écriture, demander le jour, les notes et les écrire dans l'enregistrement.

```
10 ' AGENDA
20 OPEN "D",#1,"MAI"
30   INPUT "LECTURE(1),ECRITURE(2) OU RI
EN (3)";REP
40   IF REP=0 THEN 70
50   ON REP GOSUB 100,200
60   GOTO 30
70 CLOSE #1
80 END
100 ' LECTURE
110 INPUT "QUEL JOUR";J
120 IF J>LOF(1) THEN RETURN
130 GET #1,J
140 INPUT #1,QUOI$,H$
145 IF QUOI$=SPACE$(124) THEN
    QUOI$="RIEN"
150 PRINT "LE";J;QUOI$;" A ";H$
160 RETURN
200 ' ECRITURE
210 INPUT "QUEL JOUR";J
220 IF J>LOF(1) THEN RETURN
230 INPUT "NOTER";QUOI$
235 INPUT "A QUELLE HEURE";H$
240 WRITE #1,QUOI$,H$
250 PUT #1,J
260 RETURN
```

Pour éviter l'affichage de quatre lignes blanches lorsqu'un jour est vide, il suffit d'ajouter un message :

```
145 IF QUOI$=SPACE$(124) THEN
    QUOI$="RIEN"
```


Plusieurs données dans un enregistrement

Il est encore facile d'améliorer l'agenda en notant l'heure correspondant à l'occupation inscrite :

```
235 INPUT "A QUELLE HEURE";H$
240 WRITE #1,QUOI$,H$
```

Il faut modifier en conséquence la lecture :

```
140 INPUT #1,QUOI$,H$
150 PRINT "LE";J;QUOI$;" A ";H$
```

Si vous le préférez, vous pouvez noter chaque jour l'occupation du matin et celle de l'après-midi. Il faut seulement veiller à ce que la lecture (INPUT #) des variables soit exactement la même que leur écriture (WRITE #).

Modifier la taille d'un enregistrement

Nous avons vu que la taille d'un enregistrement est celle d'une mémoire tampon (1 secteur). En effet, PUT # et GET # effectuent le transfert de *toute* la zone tampon entre la disquette et la mémoire centrale. Vous le vérifiez d'ailleurs facilement sur le catalogue : la taille du fichier "MAI" est de 31 secteurs, donc 4 blocs.
Si vous avez peu de choses à écrire dans chaque enregistrement, comme par exemple une occupation par jour, cela fait beaucoup de blancs sur la disquette, et autant de place perdue.
Vous pouvez réduire (ou augmenter) la taille d'un enregistrement. Si par exemple chaque enregistrement ne dépasse pas une ligne à l'écran, soit 40 caractères, il suffit d'ajouter 40 à la fin de l'instruction OPEN (en lecture et en écriture) :

```
OPEN "D", # 1, "AOUT", 40
```

Attention, impossible de changer de valeur pour un enregistrement : une fois ce nombre fixé, tous les enregistrements sont limités à 40 caractères.

Découper un enregistrement en plusieurs morceaux

Nous allons maintenant voir une deuxième manière (plus performante) de placer plusieurs données dans un enregistrement.
Vous êtes organisateurs des 15 km de Trifouillis-les-Bains, grand rassemblement de centaines d'adeptes de la course à pied.
Comme vous venez d'acquérir un micro-ordinateur, vous pensez qu'il peut vous aider. C'est ce que nous allons voir.
Pour vous, la première étape est d'enregistrer les candidats. Pour chaque candidat, il faut noter son nom, son prénom (vous pensiez peut-

être vous contenter du nom, mais dans ce genre de courses il y a souvent des familles entières...) et enfin son numéro de dossard. Appelons le fichier : "INSCRIPT".
Le numéro de dossard sera le numéro de l'enregistrement sur la disquette. Afin de ne pas perdre de place sur la disquette, limitons la taille d'un enregistrement à 40 caractères :

```
20 OPEN"D",#1,"INSCRIPT",40
```

Une instruction FIELD permet de diviser chaque enregistrement en zones réservées à des variables différentes :

```
30 FIELD #1,15 AS NOM$,15 AS PRENOM$
```

réserve les 15 premiers caractères de chaque enregistrement pour NOM$ et les 15 suivants pour PRENOM$ (FIELD = champ). Il reste encore 10 caractères pour inscrire ultérieurement d'autres données. Pour écrire les deux données NOM$ et PRENOM$, nous utiliserons l'instruction LSET (Left SET = placer à gauche) au lieu de WRITE # :

```
10 ' INSCRIPTION ECRITURE
20 OPEN"D",#1,"INSCRIPT",40
30 FIELD #1,15 AS NOM$,15 AS PRENOM$
40 LSET NOM$="DUMOULIN"
50 LSET PRENOM$="HENRI"
60 PUT #1,1
70 CLOSE #1
```

L'instruction LSET place le nom dans la zone de 15 caractères réservée pour NOM$, en le cadrant à gauche de la zone et en remplissant le reste de la zone avec des blancs :

DU MOULIN

Si vous inscrivez M. DUBOUT DE LA MONTAGNE, son nom sera tronqué sans aucun respect par l'ordinateur en :

DUBOUT DE LA MO

C'est à vous de faire preuve de tact !
En passant signalons que l'instruction LSET gagne 4 caractères par rapport à WRITE # (2 pour les guillemets et 2 pour le retour à la ligne).
Si la donnée à écrire est un nombre, vous pouvez la cadrer à droite de la zone réservée par l'instruction RSET (Right SET)

13

Reprenons maintenant le programme d'écriture pour que :
— il demande le nom et le prénom,

```
35 INPUT"NOM";R$
40 LSET NOM$=R$
45 INPUT"PRENOM";R$
50 LSET PRENOM$=R$
```

— il inscrive tous les coureurs (on arrêtera le programme avec un nom fictif comme "Z")

```
10 ' INSCRIPTION
20 OPEN"D",#1,"INSCRIPT",40
30 FIELD #1,15 AS NOM$,15 AS PRENOM$
33 I=1
35    INPUT"NOM";R$
37    IF R$="Z" THEN 70
40    LSET NOM$=R$
45    INPUT"PRENOM";R$
50    LSET PRENOM$=R$
60    PUT #1,I
65    I=I+1 : GOTO 35
70 CLOSE #1
```

RUN

Allez-y, inscrivez vos coureurs. Le test d'arrêt en ligne 37 est indispensable : si vous arrêtez un programme d'écriture de fichier par [CNT] [C] (ou un message d'erreur), le fichier n'est pas fermé puisque l'instruction CLOSE n'a pas été exécutée.
Vous risquez ensuite d'écrire involontairement n'importe quoi sur la disquette. Dans ce cas là, il est prudent de faire :

UNLOAD

pour fermer les fichiers restés ouverts.

Une facilité d'écriture dans PUT # et GET

Le remplissage de ce fichier à accès direct se fait de façon séquentielle : vous inscrivez le numéro 1, le 2, le 3...
(Ce n'était pas le cas dans le remplissage de l'agenda).
Dans ce cas vous n'êtes pas obligé de préciser le numéro de l'enregistrement dans l'instruction PUT # : les inscrits sont automatiquement affectés aux enregistrements successifs (le premier étant le n° 1)

Vous pouvez donc supprimer la variable de comptage I : effacez la ligne 33,

60PUT#
65GOTO 35

Pour savoir le nombre d'inscrits, c'est-à-dire le n° du dernier inscrit, ajoutez :

80 PRINT "NOMBRE D'INSCRITS" ; LOF(1)

Cette facilité d'écriture dans PUT# existe également avec GET# : si vous lisez les enregistrements les uns après les autres vous n'êtes pas obligé de préciser le numéro de l'enregistrement.

Relire les chaînes de caractères (lire les inscrits)

Si vous voulez savoir qui est inscrit sous le dossard n° 3, il suffit de lire l'enregistrement correspondant :

```
10  ' LECTURE DES INSCRITS
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$
40  GET #1,3
50  PRINT "NOM:";NOM$
60  PRINT "PRENOM:";PRENOM$
100 CLOSE #1
```

Si vous avez inscrit une centaine de noms, la lecture du 80e inscrit sera infiniment plus rapide que dans un fichier à accès séquentiel où il faut lire 79 enregistrements avant d'accéder au 80e.
Inutile dans la lecture d'un enregistrement de faire INPUT# après GET# : le contenu de la variable NOM$ qui a été déposé dans la zone tampon n° 1 par GET# est directement accessible.

Mais attention, une variable de champ définie par une instruction FIELD, comme NOM$ ne peut être modifiée que par une instruction LSET (ou RSET) : elle est toujours à la même place dans la mémoire et a une longueur fixée à l'avance (dans l'instruction FIELD).
Reprenez maintenant le programme de lecture d'un enregistrement comme nous l'avons déjà fait pour le programme d'écriture :
— demander quel est le numéro désiré (N)
— lire l'enregistrement désiré.
On arrête le programme avec un numéro inutilisé, par exemple 0.


```
10    LECTURE DES INSCRITS
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$
40     INPUT "NUMERO";N
50     IF N=0 THEN 100
60     GET #1,N
70     PRINT "NOM:";NOM$
80     PRINT "PRENOM:";PRENOM$
90     GOTO 40
100 CLOSE #1
```

RUN

Vous relisez les noms et prénoms des coureurs déjà inscrits.

Effacer le contenu d'un enregistrement (supprimer un inscrit)

Revenons à la course qui n'en est qu'à ses préparatifs. Un des candidats déclare forfait avant le départ de la course. Il faut annuler son enregistrement, par exemple en inscrivant 15 blancs pour le nom et le prénom :

```
10  ' SUPPRESSION D'UN INSCRIT
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$
40     INPUT "NUMERO A SUPPRIMER";N
50     LSET NOM$=SPACE$(15)
60     LSET PRENOM$=SPACE$(15)
70     PUT #1,N
80  CLOSE #1
```

Écrire des données numériques (enregistrer l'ordre d'arrivée)

Résultat des courses

Pour chaque coureur que l'on connaît par son numéro de dossard (on lui demandera son nom plus tard, quand il sera moins essoufflé), il faut enregistrer l'ordre d'arrivée. En même temps, on connaîtra son nom et son prénom.
Dans chaque enregistrement dimensionné à 40 caractères, il reste 10 caractères pour écrire des données supplémentaires. Modifions donc l'enre-

gistrement pour ajouter l'ordre d'arrivée de chaque coureur. Il faut (pour tous les enregistrements de 1 jusqu'à la fin du fichier) :
— entrer le numéro d'enregistrement (dossard) : N
— lire l'enregistrement N
— afficher le nom et le prénom
— déposer l'ordre d'arrivée (ARRIV) dans l'enregistrement
— réécrire l'enregistrement

```
10  ' ARRIVEE1
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$,4
AS ARRIV$
40  FOR ARRIV=1 TO LOF(1)
50    PRINT "ORDRE D'ARRIVEE:";ARRIV
55    PRINT "DOSSARD NUMERO:";N
60    GET#1,N
70    PRINT NOM$;PRENOM$
80    LSET ARRIV$=MKS$(ARRIV)
90    PUT #1,N
100 NEXT ARRIV
110 CLOSE #1
```

La ligne 80 vous étonne sans doute ! L'instruction FIELD n'accepte que les variables chaînes de caractères. Il faut donc convertir la variable numérique ARRIV en chaîne de caractères. La première idée qui vient à l'esprit est STR$ (ARRIV). C'est un moyen. Mais le Basic comporte une fonction de conversion spéciale pour déposer un nombre dans une variable destinée à une instruction FIELD. Cette fonction, MKS$( ), donne une chaîne qui occupe toujours exactement 4 octets (si le nombre est en simple précision).
Ainsi il est facile de préciser la taille à réserver pour inscrire l'ordre d'arrivée dans l'instruction FIELD : même si vous avez des dizaines de milliers de coureurs, il suffira de 4 octets pour inscrire l'ordre d'arrivée de la lanterne rouge.
Pour relire le fichier des coureurs, il faut convertir ARRIV$ en nombre : c'est CVS( ) la fonction inverse de MKS$, qui le réalise.

```
10  ' LECTURE APRES ARRIVEE
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$,4
AS ARRIV$
40      INPUT "NUMERO";N
50      IF N=0 THEN 100
60      GET #1,N
65      ARRIV=CVS(ARRIV$)
```



```
70      PRINT "NOM:";NOM$
80      PRINT "PRENOM:";PRENOM$
85      PRINT "ORDRE D'ARRIVEE:";ARRIV
90      GOTO 40
100 CLOSE #1
```

Si les nombres sont entiers (noter les temps)

Encore une amélioration, ensuite vous ferez les vôtres. Il reste un peu de place sur chaque enregistrement, profitons-en pour inscrire le temps en heures, minutes, secondes.
Il faut écrire trois nombres entiers. Le problème est le même que pour l'ordre d'arrivée : il faut convertir ces nombres en chaînes de caractères. Or il existe une fonction de conversion pour les nombres entiers : MKI$( ). Le résultat n'occupe que 2 octets.
Plaçons donc les nombres d'heures, minutes, secondes dans des variables entières : H%, M%, S%.
Les chaînes de caractères correspondantes seront : MKI$(H%), MKI$(M%), MKI$(S%).

```
10  ' ARRIVEE2
20  OPEN "D",#1,"INSCRIPT",40
30  FIELD #1,15 AS NOM$,15 AS PRENOM$,
    4 AS ARRIV$,2 AS HEURE$,2 AS MIN$,
    2 AS SEC$
40  FOR ARRIV=1 TO LOF(1)
50    PRINT "ORDRE D'ARRIVEE:";ARRIV
55    PRINT "DOSSARD NUMERO:";N
60    GET#1,N
70    PRINT NOM$;PRENOM$
80    LSET ARRIV$=MKS$(ARRIV)
82    INPUT"TEMPS(H,M,S)";H%,M%,S%
84    LSET HEURE$=MKI$(H%)
86    LSET MIN$  =MKI$(M%)
88    LSET SEC$  =MKI$(S%)
90    PUT #1,N
100 NEXT ARRIV
110 CLOSE #1
```

Ainsi les 40 octets de chaque enregistrement sont occupés. Si vous vouliez ajouter d'autres informations, il faudrait recopier tous les enregistrements dans des enregistrements plus grands.
D'où l'intérêt de bien prévoir à l'avance ce qu'on veut enregister, ou de garder un peu de marge...

Pour relire les heures, minutes, secondes, nous allons utiliser la fonction CVI( ), inverse de MKI$( ).
Le programme de lecture après l'arrivée se modifie alors de la manière suivante :

```
30  FIELD #1,15 AS HEURE$,15 AS PREHEURE$,
    4 AS ARRIV$,2 AS HEURE$,2 AS MIN$,
    2 AS SEC$
50    H%=CVI(HEURE$):M%=CVI(MIN$),
      S%=CVI(SEC$)
80    PRINT"TEMPS:";H%;"H";M%;"M";S%;"S"
```

Voilà, nos améliorations sont terminées mais vous pouvez encore en faire vous-même. S'il vous venait à l'idée d'ajouter des nombres avec une très grande précision, voici encore quelques informations (TO 7 uniquement) :
— les nombres en double précision occupent 8 octets dans une variable de FIELD,
— on les convertit en chaîne de caractères par MKD$( ),
— la conversion inverse se fait par CVD( ).
Vous savez tout sur les fichiers à accès direct.

Utilisation de LSET et RSET sur MO 5

En principe, LSET et RSET sont destinés à déposer des données dans un enregistrement. Dans le Basic Disque du MO 5, leur fonction a été étendue, en dehors de la gestion des fichiers, au transfert de données d'une chaîne à une autre chaîne quelconque.
Ainsi est-il possible d'écrire :

LSET A$ = NOM$

même si la variable A$ n'a pas été définie dans un FIELD au préalable. Cette instruction transfère le contenu de NOM$ dans A$. Ceci suppose que A$ existe déjà et que sa longueur n'est pas nulle. On peut par exemple la fixer par :

A$ = SPACE$ (15)

Si la longueur de NOM$ est inférieure à celle de A$, le contenu est cadré à gauche dans A$, le reste étant comblé par des espaces. Si la longueur de NOM$ est supérieure à celle de A$, le contenu est tronqué à droite. Si les deux longueurs sont égales, tout va bien.
Il en va de même pour RSET, à ceci près que les données sont cadrées à droite.

Ceci peut être assez pratique pour transférer rapidement des données dans une zone de taille fixe. On peut également s'en servir pour cadrer des chaînes de caractères avant sortie sur imprimante, à la manière de PRINT USING.

En *résumé* :
— dans un fichier à accès direct tous les enregistrements ont la même longueur fixée une fois pour toutes ;
— le transfert entre la mémoire centrale et la disquette se fait enregistrement par enregistrement en passant par une zone tampon ;
— l'écriture dans la zone tampon peut se faire de deux manières :

- avec WRITE #, et il faut alors faire attention à ne pas "déborder" la zone tampon avec un enregistrement. On relit alors avec INPUT #.
- avec LSET (ou RSET) : on dépose les données dans des variables de longueur déterminée par FIELD. Les variables numériques doivent alors être converties en chaînes de caractères (MKI$, MKD$ *, MKS$) et ces chaînes converties en nombres à la lecture (CVI, CVD*, CVS).

* Sur le TO 7 seulement.

Annexe 1 / Description des instructions et fonctions de gestion de fichiers sur disques avec les nouvelles instructions et fonctions du Basic Disque.

Cette annexe décrit le fonctionnement de toutes les instructions et fonctions de gestion de fichiers et toutes les nouvelles instructions et fonctions du Basic Disque. Les instructions et les détails spécifiques du TO 7 sont marquées par : (TO 7). Ce qui est spécifique du MO 5 est marqué par : (MO 5).
Avant d'entrer dans le détail de chaque instruction et de chaque fonction, rappelons quelques conventions de notations.

— les mots entre crochets [ ] représentent la partie facultative d'une instruction. Les principaux paramètres qui changent d'une utilisation à l'autre sont désignés par :

Nombre : représente une constante numérique (3*55) ou une variable numérique (Z) ou une expression dont le résultat est numérique (Z*RND)

Chaîne de caractères : représente une constante (''JULES'') ou une variable (P$) ou une expression dont le résultat est une chaîne de caractères (MID$(''JULES'', 1,1))

Donnée : représente un *nombre* ou une *chaîne de caractères*.

Variable : représente une variable numérique (X) ou une variable chaîne de caractères (Y$)

— les différents exemples fournis avec chaque instruction illustrent la partie optionnelle de l'instruction. En général, les options sont précisées dans l'explication.

Descripteur de fichier

Dans un programme ou une commande, on désigne un fichier par son descripteur. Le descripteur de fichier a la structure générale suivante :
Nom de périphérique : (options) nom de fichier

Le **nom de périphérique** peut être choisi parmi les suivants :

Ø	1er lecteur de disquette
1	2e lecteur de disquette
2	3e lecteur de disquette
3	4e lecteur de disquette
CASS	magnétophone à cassette
COMM	voie de communication série (TO 7)
KYBD	clavier
LPRT	imprimante
SCRN	écran

Le nom de périphérique n'est pas obligatoire. Sa valeur par défaut est le lecteur Ø. Parmi ces périphériques, deux d'entre eux peuvent comporter des options :

— pour l'imprimante, le nombre en option indique le nombre de caractères par ligne :
ex. : "LPRT : (8Ø)" pour une imprimante à 8Ø colonnes.

— pour la voie de communication série (TO 7), le nombre en option comporte trois parties :

- le 1re chiffre fixe la vitesse de transmission (1 = 11Ø bauds, 2 = 3ØØ, 3 = 6ØØ, 4 = 1 2ØØ, 5 = 2 4ØØ, 6 = 4 8ØØ bauds)
- le 2e chiffre fixe le nombre de bits transmis : 7 ou 8
- la suite fixe le nombre de caractères par ligne (entre 1 et 254)

ex. : "COMM : (47120)" pour une transmission à 1 2ØØ bauds sur 7 bits avec 12Ø caractères par ligne.

Le nom de fichier est constitué de deux parties séparées par un point :
nom . suffixe
Le **nom** comporte 8 caractères au plus ; toutes les lettres, les chiffres et les signes sont permis sauf le point.

Le **suffixe**, qui n'est pas obligatoire, comporte 3 caractères au plus. Quand le nom de fichier ne comprend pas de suffixe, les valeurs suivantes sont prises par défaut :

BAS : pour un programme BASIC,

DAT : pour un fichier de données,
BIN : pour un fichier binaire (zone mémoire ou programme en langage machine).

N.B. : Il existe un nom particulier réservé aux programmes dont on souhaite l'exécution directe au chargement du Basic : AUTO.BAT.

AUTO numéro, pas

Propose automatiquement les numéros de ligne pour l'écriture de programmes. Le premier nombre indique le numéro de la première ligne à écrire (1Ø par défaut). Le deuxième nombre indique l'écart entre les lignes (1Ø par défaut). Si la ligne proposée existe déjà, il s'affiche derrière le numéro un * (TO 7), ou le message "Line not empty" (MO 5). Le numéro de ligne proposé peut être modifié : AUTO se cale alors sur le nouveau numéro (MO 5). On sort de la numérotation automatique par CNT C. Le positionnement du curseur sur une autre ligne provoque aussi la sortie (TO 7).

```
AUTO 1ØØØ, 5
AUTO
```

BACKUP n° de lecteur 1 [TO n° de lecteur 2]

Sauvegarde le contenu d'une disquette sur une autre disquette. Cette commande recopie intégralement la disquette qui se trouve dans le lecteur n° 1 sur la disquette qui se trouve dans le lecteur n° 2. Avec un seul lecteur BACKUP n° recopie d'une disquette sur l'autre, en plaçant en alternance la disquette *"source"* et la disquette *"destination"* dans le lecteur.
BACKUP détruit le programme présent en mémoire centrale.

```
BACKUP Ø TO 1
BACKUP Ø        lecteur n° Ø seul.
BACKUP 1        lecteur n° 1 seul.
```

CIRCLE (colonne, ligne), rayon, couleur

Trace un cercle dont le centre est situé au point (colonne, ligne). Si la couleur n'est pas indiquée, le cercle est tracé dans la couleur courante des caractères. La couleur est une valeur numérique comprise entre − 8 et 7 (TO 7) ou entre − 16 et 15 (MO 5). Le rayon, positif, ne doit pas dépasser 255.

```
CIRCLE (16Ø, 1ØØ), 9Ø, 2.
```

CLOSE # n° de canal [, # n° de canal]

Ferme le (ou les) fichier(s) dont on indique le numéro de canal. Cette opération marque la fin du fichier et inscrit dans le catalogue toutes les informations nécessaires. Si aucun numéro de canal n'est précisé, CLOSE ferme tous les fichiers. Ne pas oublier qu'un arrêt en cours d'écriture ou de lecture par CNT C ne ferme pas les fichiers. Tous les fichiers doivent avoir été fermés avant de retirer la disquette du lecteur.

```
CLOSE #1, #3
CLOSE
```

COPY descripteur de fichier 1 [TO descripteur de fichier 2]

Permet la copie intégrale d'un fichier (en changeant éventuellement son nom) sur le même support ou sur un autre support. Avec deux lecteurs de disquettes, COPY permet de copier un fichier de l'un sur l'autre.
Avec un seul lecteur de disquettes, COPY permet de copier un fichier :
— sur la même disquette, en changeant son nom,
— sur une autre disquette, sans changer son nom.
Dans ce cas on place alternativement la disquette *source* et la disquette *destination* dans le lecteur.
On indique alors dans la commande que le nom du fichier à recopier (COPY nom de fichier).
COPY ne permet pas de transfert d'une disquette vers un autre périphérique.

```
COPY "Ø : DESS.DAT" TO "1 : DESS.DAT"
COPY "MENU.BAS" TO "CHOIX.BAS"
COPY "BUDGET.DAT"
```

CVD (variable de champ) **(TO 7)**

Dans l'utilisation d'un fichier à accès direct, assure la conversion du contenu de la variable de champ (chaîne de caractères) en un nombre en double précision. La variable de champ doit avoir été définie au préalable dans un FIELD. La conversion n'a de sens qu'après la lecture d'un enregistrement par une instruction GET#.
La variable de champ doit contenir des données qui ont été enregistrées par une conversion à l'aide de MKD$( ).
CVD n'existe pas sur MO 5.

```
A # = CVD(PRIX$)
```


CVI (variable de champ)

Cette fonction analogue à CVD( ) assure la conversion du contenu de la variable de champ en un nombre entier. La variable de champ doit correspondre à des données enregistrées par une conversion à l'aide de MKI$( ).

```
I% = CVI(Q$)
```

CVS (variable de champ)

Cette fonction analogue à CVD( ) assure la conversion du contenu de la variable de champ en un nombre en simple précision. La variable de champ doit contenir des données enregistrées par une conversion par MKS$( )

```
J = CVS(TAUX$)
```

DEF FN nom (variable [,variable]...) = expression

Définit une nouvelle fonction qui porte sur les variables indiquées entre parenthèses. Le nom de la fonction suit exactement les mêmes règles qu'un nom de variable. La partie à droite du signe = est l'expression qui définit la fonction. Elle utilise les mêmes variables que celles indiquées dans la partie gauche. Les noms de variables peuvent être identiques à ceux déjà utilisés par ailleurs, sans avoir d'interaction. La définition de la fonction doit se faire avant son utilisation. Les fonctions ainsi définies s'utilisent comme les fonctions ordinaires de BASIC. On les appelle par leur nom : FN nom (variable, variable,...). Les fonctions définies peuvent donner un résultat numérique ou un résultat chaîne de caractères.

```
DEF FN DIST (X,Y) = SQR (X ↑ 2 + Y ↑ 2)
A = FNDIST (3,5)
DEF FNCH$ (1) = MID$(STR$(1),2)
AS = FNCH$ (D$)
```

DEF USRn = adresse

Définit la fonction utilisateur numéro *n*, commençant à l'adresse mémoire indiquée. Il peut y avoir dix fonctions utilisateur simultanées, numérotées de Ø à 9. L'adresse indique le début de la routine en langage machine. Une zone mémoire destinée aux fonctions utilisateur peut être réservée par : CLEAR, adresse.
Le module en langage machine doit se terminer par l'instruction RTS du 6809 et ne doit pas modifier les contenus des registres S

et DP. Le passage des paramètres utilise, à l'appel comme au retour, les registres suivants :

Accumulateur A : (type de la variable)	2	nombre entier
	3	chaîne de caractères
	4	simple précision
	8	double précision (TO 7)
Index X : (pointeur sur la valeur)	2,X	nombre entier
	0,X	réel simple ou double précision
	0,X	descripteur de chaîne de caractères
Accumulateur B :		longueur de la chaîne de caractères
Régistre U :		adresse du 1er octet de la chaîne (MO 5 seulement)

Les nombres entiers sont codés sur 2 octets (notation en complément à deux), les réels simple précision sur 4 octets (1er octet : exposant, reste : mantisse), les réels double précision sur 8 octets (idem, TO 7 seulement)
Le descripteur de chaîne de caractères comprend 3 octets : le premier contient la longueur de la chaîne, les deux suivants l'adresse du 1er caractère.

```
DEF USR2 = &HBF8Ø
W = USR2 (V)
```

DEVICE "nom de périphérique"

Détermine le nom du périphérique qui sera pris par défaut dans un descripteur de fichier qui ne le précise pas. Au chargement du BASIC Disque, le périphérique implicite est le lecteur Ø. Les sauvegardes et chargements qui ne comportent que le nom de fichier se font sur le lecteur Ø.

```
DEVICE "1 :"
DEVICE "CASS :"
```

DIR ["numéro de lecteur :"]

Affiche à l'écran les renseignements du catalogue pour la disquette correspondante ("1" par exemple). Si rien n'est précisé, on obtient le catalogue de la disquette du lecteur pris par défaut ("Ø :" sauf modification par DEVICE).
Pour chaque fichier, les renseignements du catalogue comprennent cinq parties :
1 . nom du fichier (8 caractères)

2 . suffixe
BAS : programme BASIC
DAT : données
BIN : binaire

3 . type de fichier
B : programme BASIC
D : données BASIC
A : programme en assembleur
M : programme en langage machine

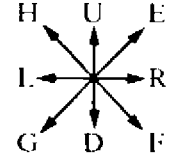
4 . type de données
A : ASCII
B : Binaire

5 . taille du fichier en nombre de blocs.
L'exécution de DIR pour un périphérique autre qu'un lecteur de disquettes provoque une erreur FC ou 5.

```
DIR "1 :"
DIR
```

DRAW chaîne de caractères.

Trace le dessin défini dans la chaîne de caractères. Les ordres élémentaires du langage graphique exécuté par DRAW sont les suivants :



U (Up), D (Down), R (Right), L (Left), E, F, G, Hsuivis d'un nombre : déplacement du nombre indiqué dans la direction fixée. Si le nombre vaut 1, on peut ne pas l'indiquer.

```
D 3Ø
```

Move : M *colonne, ligne*
déplacement au point situé en (colonne, ligne)

```
M 1ØØ, 15Ø
```

M ± C, ± L
déplacement relatif de ± C en colonnes et de ± L en lignes. Le signe est obligatoire dans ce cas.

```
M + 1Ø, - 1Ø
```

Scale : S *nombre*
changement d'échelle (de 1 à 63). L'échelle normale est 4 (rapport 1 : 1)

```
S 16
```

Arrow : A *nombre*
fixe l'orientation générale des déplacements ultérieurs, effectue une rotation sur les déplacements ultérieurs. Le nombre de Ø à 3 correspond à : Up (Ø), Right (1), Down (2), Left (3).
A 2

Color : C *couleur*
fixe la couleur du tracé, de – 8 à 7 pour TO 7, de – 16 à 15 pour MO 5.
C 3

Blank : B
placé devant un ordre effectue le déplacement sans tracer.
BM 1ØØ, 15Ø

No Update : N
placé devant un ordre permet de revenir au point de départ après un déplacement.
N U2Ø

Execute : X *variable chaîne ;*
effectue le tracé défini dans la variable chaîne de caractères.
XA$;

On peut introduire la valeur d'une variable numérique par :
= *variable* ;
U=NB ;

Les ordres élémentaires peuvent être séparés par un point-virgule ou un espace. On peut également mettre un espace entre un ordre et son argument.

```
DRAW "C1;BM115Ø,15ØU20R2ØG2Ø"
DRAW "U=NB1 ; L=NB2 ;"
DRAW "U3Ø R3Ø G3Ø"
DRAW "U3Ø"
```

DSKI$ (n° de lecteur, piste, secteur)

Cette fonction effectue la lecture d'un secteur pour lequel on donne le numéro, le numéro de la piste et celui du lecteur où il se trouve. Le résultat est une chaîne de caractères de 128 octets. Le numéro de lecteur est compris entre Ø et 3 ; le numéro de piste entre Ø et 39, le numéro de secteur entre 1 et 16 inclus. Toute tentative de lecture en dehors de ces valeurs provoque une erreur FC ou 5.

DSKI$ (Ø,2Ø,2)

DSKINI n° de lecteur [, facteur d'entrelacement]

Initialise la disquette qui se trouve dans le lecteur précisé. La disquette ne doit pas être protégée en écriture, c'est-à-dire que son encoche de protection doit être libre. Après initialisation, le catalogue de la disquette est vide et le nombre de blocs libres est de 78. L'initialisation d'une disquette déjà enregistrée provoque son effacement intégral et toutes les données qu'elles contenait sont perdues. Si la disquette est défectueuse, le message IO Error ou 53 Error apparaît. Le facteur d'entrelacement, qui n'est pas obligatoire, permet de choisir le nombre de secteurs compris entre deux secteurs de numéros consécutifs (implicitement fixé à 4 en l'absence de précision). Le numéro du lecteur est obligatoire. Si l'on a fait au préalable VERIFY ON, l'initialisation a lieu avec vérification et dure beaucoup plus longtemps.

DSKINI Ø

DSKF (n° de lecteur)

Cette fonction donne le nombre de blocs libres sur la disquette située dans le lecteur correspondant. Cette valeur s'étend de 78 (disquette vide) à Ø. Un bloc est un paquet de 8 secteurs consécutifs et peut contenir un volume de données de 1K octets soit 1024 caractères.

DSKF (Ø)

DSKO$ n° de lecteur, piste, secteur, chaîne de caractères

Cette instruction dépose la chaîne de caractères dans le secteur dont on donne le numéro, le numéro de la piste et le numéro du lecteur sur lequel il se trouve. Le numéro du lecteur doit être compris entre Ø et 3, le numéro de la piste entre Ø et 39, le numéro du secteur entre 1 et 16. La chaîne de caractères doit avoir une taille de 128 caractères au maximum. Si sa taille est inférieure, le secteur sera rempli de caractères de code ASCII Ø dans la partie vide après le dernier caractère.

DSKO$ Ø,18,3,HA$

EOF (n° de canal)

Cette fonction retourne – 1 (VRAI) si la fin du fichier est atteinte ou Ø (FAUX) sinon.

EOF (1)

FIELD [#] n° de canal, longueur 1 AS variable de champ 1 [,longueur 2 AS variable de champ 2]...

Définit l'organisation en champs de l'enregistrement du fichier à accès direct dont on donne le numéro de canal. Pour chaque champ, on précise sa longueur en caractères et le nom de la variable de champ qui le désigne. Cette variable est toujours une variable chaîne de caractères. La somme des longueurs ne doit pas dépasser la longueur de l'enregistrement du fichier qui a été fixée au moment de l'ouverture (OPEN). Si la longueur n'a pas été fixée dans OPEN elle est de 128 caractères. L'instruction FIELD est en général exécutée une fois, après l'ouverture du fichier par OPEN. Tous les enregistrements sont alors lus et écrits suivant ce format. Cependant il est possible d'exécuter une instruction FIELD différente pour lire des enregistrements de formats différents.
Les variables de champ ainsi définies ne peuvent pas être manipulées comme n'importe quelle variable. Leur contenu est toujours utilisable pour être affiché, transformé ou affecté à une autre variable. Par contre, on ne peut y déposer une valeur que par les instructions LSET et RSET. Ces variables chaînes de caractères peuvent également recevoir des nombres après que ceux-ci aient été convertis par les fonctions MKIS (nombres entiers), MKS$ (nombres décimaux en simple précision), MKD$ (nombres sur TO 7 en double précision). La taille d'un nombre converti par ces fonctions est de :
— 2 octets s'il est entier
— 4 octets en simple précision
— 8 octets en double précision

```
FIELD # 2, 15 AS NOM$, 4 AS Q$
```

FILES nombre de canaux, longueur

Cette instruction réserve le nombre de canaux utilisables simultanément dans un programme. Ce nombre ne peut pas être supérieur à 16. Chaque canal réservé occupe une zone de 153 octets en mémoire pour les informations concernant le fichier et la zone de transit des enregistrements. Au chargement du Basic Disque, le nombre de canaux réservés est fixé à 2. Si l'instruction FILES comporte une indication de longueur (celle-ci n'est pas obligatoire), cette longueur fixe la somme des longueurs des enregistrements des fichiers à accès direct ouverts simultanément. Elle est fixée à 256 octets au chargement du Basic Disque (2 fichiers aux enregistrements de 128 octets).

Par exemple, pour pouvoir ouvrir trois fichiers à accès direct dont les longueurs d'enregistrement sont respectivement 30, 40 et 50 caractères, il faut réserver une longueur d'au moins 120 caractères. L'instruction FILES remet à zéro toutes les variables existantes. Elle doit donc être exécutée en mode direct ou en tout début de programme.

```
FILES 3, 120
FILES 1
```

GET (colonne 1, ligne 1) – (colonne 2, ligne 2), tableau

Mémorise dans un tableau numérique la portion d'image graphique contenue dans le rectangle défini par les deux sommets (colonne 1, ligne 1) et (colonne 2, ligne 2). Le tableau numérique (entier, simple ou double précision) doit avoir été déclaré au préalable et sa dimension suffisante pour recevoir l'image, compte tenu du fait que un octet peut recevoir deux points de l'image (TO 7), ou un seul point (MO 5), et qu'un élément de tableau comporte 2, 4 ou 8 octets suivant qu'on est en entiers, simple ou double précision. La portion d'image ainsi mémorisée peut être restituée en un autre endroit de l'écran par PUT.

```
DIM T (312) ou DIM T (625)
GET (150,80) – (190,140), T
```

GET # n° de canal, n° d'enregistrement.

Transfère un enregistrement d'un fichier à accès direct (dont on donne le numéro de canal) vers la zone tampon en mémoire centrale. La récupération du contenu de l'enregistrement peut alors se faire par INPUT #, ou directement dans les variables de champ si un FIELD a été exécuté au préalable. Si le numéro d'enregistrement n'est pas précisé, GET lit l'enregistrement suivant du fichier ou le 1er si aucun enregistrement n'a encore été lu ou écrit.

HEX$ (nombre)

Cette fonction donne une chaîne de caractères représentant le nombre dans la base hexadécimale (16). Sa valeur ne doit pas dépasser 65535.

```
HEX$ (65535)
```

INPUT # n° de canal, liste de variables

Cette instruction lit le fichier dont on précise le n° de canal et affecte les valeurs lues aux variables mentionnées dans la liste. Le type des données (numérique ou chaîne de caractères) doit correspondre au type des variables. Cette instruction obéit aux mêmes règles que INPUT à partir du clavier.

```
INPUT # 2, A$, N
```

INPUT$ (nombre de caractères, # n° de canal)

Cette fonction retourne une chaîne de caractères qui contient le nombre de caractères lus à la suite dans le fichier indiqué. Cette fonction obéit aux mêmes règles que INPUT$ à partir du clavier. Elle permet de connaître de façon exacte (au caractère près), le contenu d'un fichier.

```
(INPUT$ (3, #1)
```

KILL descripteur de fichier

Supprime le fichier en question, c'est-à-dire, supprime ses références dans le catalogue et libère la place qu'il occupe sur la disquette. Dans la description du fichier, le nom du périphérique n'est pas obligatoire mais le suffixe est obligatoire. Le périphérique pris par défaut est le lecteur Ø ou celui fixé par DEVICE. L'instruction KILL ne fonctionne que pour un fichier sur disquette. Si le fichier n'existe pas, le message NE Error ou 62 Error apparaît.

```
KILL "1 : BONSOIR.BAS"
KILL "RESUL.DAT"
```

LINE INPUT # n° de canal, variable chaîne

Lit le prochain enregistrement du fichier dans une variable chaîne de caractères sans tenir compte des virgules.

```
LINE INPUT # 2, X$
```

LIST descripteur de fichier, n° ligne 1 - n° ligne 2

Envoie le texte du programme dans le fichier indiqué, du numéro de ligne 1 au numéro de ligne 2 inclus. Si les numéros de lignes ne sont pas mentionnés, tout le programme est listé. Si le fichier est un fichier sur disque, LIST est équivalent à SAVE avec l'option ,A.

```
LIST "LPRT : (80)", 1000-1990
LIST "1 : BONJOUR.BAS"
LIST
```


LOAD descripteur de fichier, R

Charge en mémoire le programme dont on a précisé le nom et le support. Avec l'option ,R, le programme chargé est aussitôt exécuté et les fichiers déjà ouverts avant le chargement restent ouverts. Le périphérique pris par défaut est le lecteur Ø, ou celui fixé par DEVICE.

```
LOAD "CASS : OTHELLO",R
LOAD "TICTAC"
```

LOADM descripteur de fichier, translation ,R

Charge une zone mémoire en binaire à partir d'un fichier constitué par SAVEM. Si une translation est indiquée, les données, ou le programme sont décalés de cette translation. Si la zone contient un programme, l'option R en provoque l'exécution immédiate.

```
LOADM "HORLOGE", &H200,R
LOADM "FORME"
```

LOC (n° de canal)

Cette fonction indique où en est la lecture (ou l'écriture) dans le fichier ouvert sous le n° de canal précisé. Si le fichier est à accès séquentiel, LOC indique le nombre de secteurs écrits ou lus depuis l'ouverture du fichier. Si le fichier est à accès direct, LOC indique le numéro de l'enregistrement qui suit celui qui vient d'être lu par GET ou écrit par PUT.

```
LOC (2)
```

LOF (n° de canal)

Cette fonction donne la position du dernier enregistrement dans le fichier ouvert sous le n° de canal considéré. Si le fichier est à accès séquentiel, LOF indique le numéro du dernier secteur du fichier. Si le fichier est à accès direct, LOF indique le numéro du dernier enregistrement.

```
LOF (1)
```

LSET variable de champ = chaîne de caractères

Dépose une chaîne de caractères dans une variable de champ préalablement définie par une instruction FIELD. La chaîne de caractères est déposée en commençant par la gauche. Si sa longueur est plus petite que la taille de la variable de champ, le reste est complété à droite par des espaces ; si sa longueur est trop grande, la fin de

la chaîne est éliminée. Les nombres peuvent être déposés dans une variable de champ après avoir été convertis par :

MKI$( ) pour un entier
MKS$( ) pour un nombre en simple précision
MKD$( ) pour un nombre en double précision

Sur MO 5, il n'est pas nécessaire que la chaîne qui reçoit ait été définie par un FIELD. LSET autorise les transferts d'une chaîne quelconque à une variable chaîne quelconque.

```
LSET ID$ = NOM$ + PR$
LSETQ$ = MKS$ (QU)
```

MERGE descripteur de fichier, R

Charge en mémoire le programme indiqué et l'ajoute au programme résidant. Les lignes du programme chargé sont classées avec celles du programme résidant et les recouvrent si elles sont identiques. Le programme à charger doit avoir été sauvegardé avec l'option A. L'exécution du programme global est lancée à la fin du chargement avec l'option R.

```
MERGE "1 : TEST",R
MERGE "MENU"
```

MID$ (chaîne de caractères 1, début, longueur) = chaîne de caractères 2

Cette instruction, différente de la fonction du même nom, modifie la chaîne 1 par la chaîne 2 située à droite du signe = . Le premier nombre indique le début de la modification, le second en indique la longueur. La longueur de la chaîne 1 reste inchangée ; il ne s'opère qu'une substitution partielle. Si la chaîne modifiante (située à droite) est trop longue, elle est tronquée à droite.

```
MID$ (A$,1Ø,3) = "ABC"
```

MKD$ (nombre) **(TO 7)**

Cette fonction assure la conversion d'un nombre en double précision en chaîne de caractères, pour qu'il puisse être déposé par LSET ou RSET dans une variable de champ. La longueur de la chaîne de caractères obtenue est de 8 octets.
Cette chaîne de caractères est codée et non lisible directement. Cette fonction n'existe pas sur MO 5.

```
MKD$ (TOTAL #)
MKD$ (1.2345678Ø1234)
```


MKI$ (nombre)

Cette fonction assure la conversion d'un nombre entier en chaîne de caractères, pour qu'il puisse être déposé par LSET ou RSET dans une variable de champ. La longueur de la chaîne obtenue est de 2 octets.

```
MKI$ (Q %)
MKI$ ( – 12536)
```

MKS$ (nombre)

Cette fonction assure la conversion d'un nombre en simple précision en chaîne de caractères pour qu'il puisse être déposé par LSET ou RSET dans une variable de champ. La longueur de la chaîne obtenue est de 4 octets.

```
MKS$ (X)
MKS$ (6.Ø23 E + 23)
```

MOTOR ON / MOTOR OFF

Comme son nom l'indique, met en marche le moteur du magnétophone à cassette ou l'arrête.

```
MOTORON
```

NAME descripteur de fichier 1 AS descripteur de fichier 2.

Permet de changer le nom d'un fichier sur disquette. Descripteur du fichier 1 comporte le nom du fichier à renommer ; descripteur de fichier 2 contient le nouveau nom du fichier. Le nouveau nom ne doit pas correspondre à un fichier déjà existant.

```
NAME "1:TIC.BAS" AS "1:TAC.BAS"
NAME "TEXTE.DAT" AS "OLDTXT.DAT"
```

OPEN "I", # n° de canal, descripteur de fichier.

Ouvre le fichier séquentiel en lecture sous le n° de canal indiqué. Toutes les opérations de lecture pour ce fichier devront se faire sous ce n° de canal. Le n° de canal doit être compris entre 1 et 16. Un même fichier peut être ouvert en lecture sous plusieurs canaux simultanément. Le périphérique pris par défaut est le lecteur Ø ou celui fixé par DEVICE.

```
OPEN "I", # 2, "1 : RESUL.DAT"
```

OPEN "O", # n° de canal, descripteur de fichier

Ouvre le fichier séqentiel indiqué en écriture sous le n° de canal précisé. Le numéro de canal doit être compris entre 1 et 16. Un même fichier ne peut être ouvert qu'une fois en écriture. Sur disque, l'ouverture déclenche l'initialisation du fichier ; si un fichier existait déjà sous le même nom, il est supprimé. Le périphérique pris par défaut est le lecteur Ø ou celui fixé par DEVICE

```
OPEN "O",# 1, "CASS : TELEPH.DAT"
OPEN "O",# 3, "RESUL"
```

OPEN "D", # n° de canal, descripteur de fichier [, longueur]
OPEN, "R", # n° de canal, descripteur de fichier [, longueur]

Ouvre le fichier à accès direct dont le nom est indiqué, en lecture et en écriture, sous le n° de canal précisé. Le n° de canal doit être compris entre 1 et 16. La longueur, optionnelle, fixe la longueur de l'enregistrement. Cette longueur doit être inférieure ou égale à la longueur indiquée dans FILES. Si la longueur n'est pas précisée dans OPEN, l'enregistrement occupe alors 1 secteur soit 128 octets. Le périphérique pris par défaut est le lecteur Ø ou celui fixé par DEVICE. Ou ne peut ouvrir de fichier à accès direct que sur disquette. Les deux formes OPEN "D", et OPEN "R",.. sont équivalents.

```
OPEN "D", # 2, "1 : AGENDA.OLD", 32
OPEN "R", # 1, "RESUL"
```

PAINT (colonne, ligne) [, couleur] [, moiré]

Remplit une courbe fermée en partant du point situé en (colonne, ligne) avec la couleur précisée. La couleur doit être la même que celle de la courbe qui délimite la surface à "peindre". Cette couleur peut être de forme ou de fond (de − 8 à 7 sur TO 7 ou TO 7-70 et de − 16 à + 15 sur MO5). Si la couleur n'est pas précisée, c'est la couleur courante qui est prise par défaut. La courbe qui délimite le contour doit être parfaitement fermée. Dans le cas contraire, PAINT remplit tout l'écran. Son action peut-être arrêtée par la touche d'initialisation ou par les touches STOP ou CNT C (ces deux dernières possibilités sur MO5 seulement). Si le point est choisi à l'extérieur d'une courbe fermée, c'est l'extérieur qui est peint.

Un ordre PAINT à partir d'un point d'une couleur identique à celle choisie reste sans effet.

Le dernier paramètre, moiré, qui devra avoir pour valeur Ø ou 1 permet de peindre avec moirage si la valeur donnée est 1. Dans ce cas un point sur deux seulement est allumé, ce qui donne un effet de grisé. (Ce dernier paramètre n'existe que sur MO 5). Si ce paramètre est omis, c'est la valeur courante qui est prise par défaut. Sa valeur initiale est Ø.

```
PAINT (100,125), 3
PAINT (30,30)
PAINT (30,30), 2, 1
```

PRINT # n° de canal, donnée [, donnée,...]

Écrit les données (variables ou constantes) dans le fichier dont on précise le n° de canal. Les données sont enregistrées de la même manière qu'à l'écran avec PRINT (espaces, séparateurs,..)
Le fichier peut être séquentiel ou à accès direct. Dans ce dernier cas, PRINT # écrit dans le tampon d'enregistrement. Le transfert de l'enregistrement dans le fichier doit se faire par PUT #.

```
PRINT # 2, A$, N
PRINT # 1, "ARTHUR"
```

PRINT # n° de canal, USING chaîne de caractères ; donnée [,donnée,...]

Fonctionne comme PRINT #, mais les données sont écrites suivant le format précisé par USING. Les données sont enregistrées de la même manière qu'à l'écran avec PRINT USING.

```
PRINT # 1, USING "# # # . # #"; 120.5
```

PUT (colonne 1, ligne 1) − (colonne 2, ligne 2), tableau

Restitue à l'écran l'image contenue dans le tableau, dans le rectangle dont on donne deux sommets opposés : colonne 1, ligne 1 et colonne 2, ligne 2. L'image mémorisée dans le tableau numérique par une instruction GET préalable, doit correspondre aux mêmes dimensions (nombres de points en colonne et en ligne).

```
PUT (100,50) - (120,80), IM
```

PUT # n° de canal, n° d'enregistrement

Transfère un enregistrement d'un fichier à accès direct (dont on donne le n° de canal) de la zone tampon vers la disquette. Le contenu de l'enregistrement doit avoir été déposé par WRITE # ou PRINT #, ou directement dans les variables de champ par LSET

ou RSET. Si le numéro d'enregistrement n'est pas précisé, PUT # écrit dans l'enregistrement suivant du fichier, ou dans le 1er si aucun enregistrement n'a été lu ou écrit.

PUT # 1, 12 : PUT # 2

RENUM nouveau numéro, ancien numéro, ancien numéro, pas
1re ligne 1re ligne dernière ligne

Rénumérote un programme ou une partie de programme. Toutes les indications sont optionnelles et possèdent des valeurs par défaut. Le premier nombre indique le nouveau numéro (10 est pris par défaut). Le deuxième nombre indique à partir de quel numéro (ancien) il faut commencer à renuméroter (par défaut, c'est la première ligne du programme). Le troisième nombre indique l'ancien numéro de ligne de fin de la renumérotation (par défaut, la fin du programme). Le quatrième nombre indique l'écart entre chaque nouvelle ligne (par défaut 10). RENUM change en conséquence et dans tout le programme, tous les numéros des lignes renumérotées dans les branchements et les appels de sous-programmes (GOTO, GOSUB, THEN, ON... GOTO, ON... GOSUB, ONPEN... GOTO, ON PEN... GOSUB). Si dans la ligne L1, (ancien numéro), il est fait appel à une ligne L2 inexistante, le message suivant apparaît :
Undefined line L2 in L1
dans lequel L1 est le numéro de ligne renumérotée.
Les virgules sont indispensables, sauf les dernières.
La renumérotation d'une partie de programme n'est possible que si les nouvelles lignes restent à l'intérieur de l'espace laissé par les lignes inchangées : le nouveau n° de la 1re ligne renumérotée doit rester plus grand que le n° de la ligne inchangée qui le précède, et, le nouveau n° de la dernière ligne renumérotée doit rester plus petit que le n° de la ligne inchangée qui suit.

RENUM 1500, 1000, 1340, 5
RENUM 100, 10
RENUM

RSET variable de champ = chaîne de caractères

Dépose une chaîne de caractères dans une variable de champ préalablement définie par une instruction FIELD. La chaîne de caractères est déposée en commençant par la droite. Si sa longueur est plus petite que la taille de la variable de champ, le reste est complété à gauche par des espaces ; si sa longueur est trop grande, la fin

de la chaîne est éliminée. Les nombres peuvent être déposés dans une variable de champ après avoir été convertis par :

MKI$( ) pour un nombre entier
MKS$( ) pour un nombre en simple précision
MKD$( ) pour un nombre en double précision(TO 7).

Sur MO 5, il n'est pas nécessaie que la chaîne qui reçoit ait été définie par un FIELD. RSET autorise les transferts d'une chaîne à une variable chaîne quelconque préalablement définie.

RSET ID$ = NOMS + PRS
RSET Q$ = MKI$ (1234)

RUN descripteur de fichier, R

Lance l'exécution, après chargement, du programme dont on a précisé le nom et le support. Avec l'option R, les fichiers déjà ouverts avant le lancement restent ouverts.

RUN "1 : OTHELLO", R
RUN "TICTAC"

SAVE descripteur de fichier
SAVE descripteur de fichier, A
SAVE descripteur de fichier, P

Sauvegarde sous le nom et sur le support indiqués, le programme présent en mémoire centrale. Sur disque, si un fichier existait déjà sous le même nom, il est remplacé par le nouveau programme. Avec l'option A, le programme est sauvegardé sous une forme non codée, identique au listage à l'écran. Il pourra alors être utilisé par MERGE. Avec l'option P (protégé), le programme est sauvegardé sous une forme codée. Au chargement suivant, le programme ne pourra être ni listé ni modifié. Il est conseillé de le sauvegarder également sous la forme habituelle.

SAVE "CASS : OTHELLO", P
SAVE "TICTAC", A
SAVE "TOE"

SAVEM descripteur de fichier, adresse 1, adresse 2, adresse 3

Sauvegarde une partie de la mémoire dans un fichier. L'adresse 1 indique le début de la zone à sauvegarder, adresse 2 en indique

la fin, adresse 3 indique l'adresse de lancement de l'exécution au moment du chargement par LOADM avec l'option R. SAVEM peut servir à conserver une zone mémoire précise ou un programme en binaire.

SAVEM "HORLOGE", &H 8800, &H 8900, &H 8800

SEARCH chaîne de caractères, n° ligne 1 - n° ligne 2 (MO 5)

Affiche sur l'écran toutes les lignes du programme comportant la chaîne de caractères spécifiée, de la ligne 1 à la ligne 2 incluses. La chaîne de caractères à rechercher peut être donnée dans une constante ou une variable. Ou peut effectuer la recherche sur tout ou partie du programme, comme dans LIST.
N'existe que sur MO 5.

SEARCH "NOM$", 1000 – 2000
SEARCH "NBJ", 1000-
SEARCH "520"

SKIPF ["nom de fichier"]

Sur la cassette, saute par- dessus le fichier indiqué, ou le fichier suivant si rien n'est indiqué.

SKIPF "TICTAC"
SKIPF

SPACE$ (nombre)

Cette fonction donne une chaîne de caractères composée d'autant d'espaces que le nombre le précise.

SPACE$ (30)

STRING$ (nombre, caractère)

Cette fonction donne une chaîne de caractères identiques au caractère indiqué et dont la longueur est égale au nombre indiqué. Le caractère peut être précisé par son code ASCII.

STRING$ (10,"*")
STRING$ (20,42)

SWAP variable 1, variable 2

Cette instruction échange le contenu de deux variables de même type.

SWAP I,J
SWAP IX$, TY$

UNLOAD n° de lecteur

Ferme tous les fichiers ouverts sur la disquette située dans le lecteur dont on précise le numéro. Par défaut, UNLOAD considère le lecteur n° 0 ou celui défini dans DEVICE.
Cette instruction garantit le retrait de la disquette du lecteur sans aucun risque, en particulier après l'arrêt d'un programme d'écriture de fichiers par CNT C.

UNLOAD 1
UNLOAD

VERIFYON
VERIFYOFF

Avec l'option ON, toute écriture sur disquette sera suivie d'une vérification. L'option OFF supprime cette vérification.

VERIFY ON

WRITE # n° de canal, donnée [,donnée]...

Écrit dans le fichier dont on précise le n° de canal, la liste des données. Les chaînes de caractères sont entourées de guillemets et les différentes données sont séparées par des virgules. Après la dernière donnée, WRITE # écrit deux caractères : le saut de ligne et le retour de ligne.
Dans le cas d'un fichier à accès direct, WRITE # écrit dans le tampon d'enregistrement. Le transfert de l'enregistrement dans le fichier doit alors se faire par PUT #.

WRITE #2, NOM$, TAUX

Annexe 2 / Organisation d'une disquette

Cette annexe est destinée à ceux qui veulent en savoir un peu plus sur la manière dont le BASIC Disque gère une disquette. Les informations que nous vous donnons ici sont un peu plus techniques (il faut bien utiliser un peu l'hexadécimal pour représenter ce qui se trouve dans les octets), mais sachez bien que leur connaissance n'est pas du tout nécessaire, même pour faire un bon usage des fichiers.

Organisation physique

Une disquette est constituée de 40 pistes concentriques de 16 secteurs chacune. Chaque secteur contient 128 octets utiles. Les 40 pistes sont numérotées de 0 à 39 et les 16 secteurs de 1 à 16.
Le découpage des pistes en 16 secteurs est fait au moment de l'initialisation de la disquette par le logiciel du BASIC Disque. On dit alors que la sectorisation est logicielle ; ce qui va de pair avec le fait que la disquette ou plus exactement le support en mylar ne possède qu'un seul trou d'index sur la circonférence. Ce trou unique sert à repérer la position du 1er secteur de chaque piste.
Contrairement à ce qu'on pourrait penser à priori, le secteur n° 2 n'est pas celui qui suit immédiatement le secteur n° 1 sur la piste. En fait, ils sont séparés par 4 secteurs. De même, le secteur n° 3 est séparé du n° 2 par 4 secteurs, et ainsi de suite. Les secteurs sont donc entrelacés sur la piste avec exactement 4 secteurs entre deux numéros de secteurs successifs. Cette disposition a été adoptée pour permettre un minimum de traitement entre deux lectures ou deux écritures de secteurs successifs. En effet, il est très rare que l'ordinateur lise deux secteurs sans rien faire entre les deux lectures. Lors du chargement d'un programme par LOAD, par exemple, après avoir lu un secteur, il faut ranger la partie de programme qu'il contient dans la zone mémoire qui suit ce qui a été chargé juste avant, puis lire le secteur suivant.., etc. Le facteur 4 a été choisi pour permettre la meilleure efficacité lors de l'enregistrement et du chargement d'un programme. Toutefois, si vous estimez qu'un autre facteur serait plus efficace pour les applications que vous envisagez, il est possible de le préciser au moment de l'initialisation :

```
DSKINI 0,2
```

initialisera la disquette située dans le lecteur n° 0, en laissant 2 secteurs entre deux numéros de secteurs consécutifs.
Cette possibilité n'est vraiment intéressante que si le temps est pour vous

un élément crucial. En effet, il vous faudra vous livrer à des mesures précises avant de pouvoir déterminer quel est le facteur optimal pour chacun de vos usages.

Le catalogue

Pour des raisons de commodité d'accès, le catalogue est situé sur la piste 2Ø de la disquette, c'est-à-dire au milieu. On diminue ainsi la distance entre les données d'un fichier et les informations générales qui le concernent et qui sont inscrites dans le catalogue.
Le catalogue commence au secteur 3 et occupe une place de 14 secteurs. Chaque fichier inscrit au catalogue a une zone réservée fixe de 32 octets. Cette zone contient les informations suivantes :

Nombre d'octets	*Contenu*
8	Nom du fichier, cadré à gauche, complété au besoin par les blancs à droite. Si le fichier a été supprimé par KILL, le premier octet contient Ø. Si la zone du catalogue n'a pas encore été utilisée, le premier octet contient FF (hexadécimal).
3	Suffixe du nom du fichier, cadré à gauche, complété par des blancs à droite si nécessaire.
1	Type du fichier : Ø : programme BASIC B 1 : fichier de données BASIC D 2 : programme en langage machine M 3 : fichier de texte. A (3e colonne de DIR)
1	Type des données : Ø : les octets contiennent du binaire FF : les octets contiennent des caractères codés en ASCII (4e colonne de DIR)
1	le n° du premier bloc attribué au fichier
2	le nombre d'octets occupés dans le dernier secteur du fichier.
16	inutilisés.

Toutes les informations écrites dans le catalogue sont entièrement gérées par le BASIC. En particulier, le nombre d'octets occupés dans le dernier secteur ne peut être écrit qu'au moment de la fermeture : une bonne raison pour ne pas oublier CLOSE. Enfin, sachant qu'un fichier occupe 32 octets dans le catalogue et que ce dernier occupe 14 secteurs, combien de fichiers peuvent être écrits sur une disquette ?

La table d'allocation mémoire de la disquette

Nous avons vu avec l'instruction DIR que la mémoire allouée à chaque fichier ne se comptait pas en secteurs mais en blocs de 8 secteurs soit 1 K octets.
Le secteur n° 2 de la piste 2Ø contient la table d'allocation mémoire, c'est-à-dire des informations sur les blocs de la disquette.
Dans cette table, chaque bloc est représenté par un octet. La valeur de l'octet donne des indications sur l'occupation du bloc correspondant (1) :

Valeur (décimale)	*Signification*
entre Ø et 79	Le bloc fait partie d'un fichier, l'octet contient le n° du prochain bloc du même fichier.
entre 193 et 199	Le bloc est le dernier d'un fichier, l'octet contient le nombre de secteurs du bloc occupés par le fichier (plus 192).
254	Le bloc est réservé et ne peut pas être utilisé pour un fichier.
255	Le bloc est libre.

Cette table d'allocation est mise à jour au fur et à mesure de l'écriture dans un fichier. Cependant, elle n'est pas mise à jour à *chaque* écriture dans un fichier. C'est la raison pour laquelle il est obligatoire d'utiliser UNLOAD lorsqu'on a arrêté une opération d'écriture de façon impromptue sans qu'il y ait eu de fermeture du fichier. UNLOAD recopie cette table sur la disquette et vous évite ainsi des ennuis si vous la remplacez par une autre dans le lecteur.

(1) Attention : le 1er octet du secteur n'est pas utilisé. Le 1er bloc est représenté par le 2e octet, le 2e bloc par le 3e octet, etc.

Comment voir le contenu de la disquette ? DSKI$ et DSKO$

L'envie vous est peut-être venue de voir un peu plus précisément ce que contient la disquette, secteur par secteur. Vous pouvez le faire avec une fonction qui est analogue de PEEK( ) dans la mémoire centrale :

DSKI$(Ø, 1Ø, 3)

vous donne le contenu (une chaîne de 128 caractères) du secteur 3 de la piste 1Ø du lecteur Ø. Avec une telle fonction, vous pourrez examiner à loisir comment sont enregistrées les données dans un fichier à ac ès direct.
Cette fonction a son pendant DSKO$ qui vous permet d'écrire ce qu'il vous plaît dans le secteur que vous choisissez. Aussi, pour écrire la chaîne de caractères SCT$, de longueur 128 octets maximum, dans le secteur 3 de la piste 1Ø du lecteur Ø, vous faites :

DSKO$ Ø, 1Ø, 3, SCT$

En utilisant la fonction DSKI$( ) ou l'instruction DSKO$, n'oubliez pas que les pistes sont comptées de Ø à 39 et les secteurs de 1 à 16.
Un dernier conseil : n'employez pas DSKO$ sur une disquette qui contient des programmes ou des données qui vous sont chers, une fausse manœuvre pourrait rapidement lui être fatale. Entraînez-vous d'abord sur des copies, c'est moins risqué !

Exécution automatique d'un programme

Si vous utilisez souvent le même programme, il est agréable de pouvoir en lancer l'exécution immédiatement après le chargement du Basic Disque.
Pour ce faire, il vous faut sauvegarder le programme en question sous le nom « AUTO.BAT » sur une disquette contenant déjà le Basic Disque, c'est-à-dire sur une copie de la disquette système.
A la mise en route, quand vous choisirez l'option 2 UN PROGRAMME ENREGISTRÉ (TO 7), ou quand vous taperez la commande DOS (MO 5), le micro-ordinateur va charger d'abord le Basic Disque, puis chercher sur la disquette le programme AUTO.BAT, le charger et l'exécuter.
Pratique, non !

Annexe 3 / Les fonctions USR en langage machine

Cette annexe intéresse surtout ceux qui ont quelques notions de programmation du microprocesseur 6809 en langage machine. Elle suppose une connaissance minimale de ce microprocesseur.
En dehors des fonctions utilisateur que nous allons examiner ici, il existe déjà une solution pour programmer des utilitaires en langage machine et les exécuter.

L'instruction EXEC

La première chose à faire avant tout est de réserver la place nécessaire aux instructions machine. On utilise pour cela CLEAR qui réserve la zone mémoire utile :

CLEAR &HBEFF
: fixe l'adresse la plus haute du BASIC à &HBEFF. Il reste donc 256 octets libres jusqu'au fond de la mémoire en &HBFFF (TO 7)

La deuxième est d'y déposer les instructions par des POKE, ou bien par LOADM si ces instructions ont été conservées en fichier, prêtes à être exécutées.
Enfin, on exécute le module en langage machine par :

EXEC &HBFØØ
: si l'adresse de début du module est en &HBFØØ.

Toutefois, avant d'écrire le module en langage machine, il faut respecter certaines règles si l'on veut revenir à la suite dans le programme BASIC :
— le module doit se terminer par RTS,
— les registres S (Stack) et DP (Direct Page) ne doivent pas avoir été modifiés. Pour pouvoir les utiliser dans le module, il faudra les sauvegarder dans la pile à l'entrée et les récupérer à la sortie.

Les fonctions USRn

On peut définir 10 fonctions utilisateur en langage machine par l'instruction :

DEF USRn = adresse
n est un chiffre (de 0 à 9)
adresse désigne l'adresse mémoire à laquelle commence la fonction.

L'appel de la fonction se fait alors comme pour une fonction normale :

variable = USRn (expression)

L'avantage principal des fonctions USR est qu'elles permettent de passer des valeurs et d'en retourner au programme BASIC par le résultat fourni.
Nous allons examiner comment ces valeurs sont passées aux fonctions utilisateur.
A l'appel de la fonction,
l'accumulateur A contient le type de la donnée :
2 nombre entier
3 chaîne de caractères
4 nombre en simple précision
8 nombre en double précision (TO 7)

le registre d'index X pointe sur la valeur :
2,X nombre entier
0,X nombre en simple ou double précision
0,X descripteur pour une chaîne

Dans le cas où la donnée est une chaîne de caractères, l'accumulateur B contient la longueur de la chaîne, le registre U pointe sur le 1er caractère de la chaîne (MO 5).
Il faut remarquer que, sauf pour les chaînes de caractères, l'accumulateur A contient en même temps la longueur de la donnée en octets.
Rappelons que :
— les entiers sont codés sur 2 octets : 1er octet poids forts, 2e octet poids faibles,
— les nombres en simple précision sont codés sur 4 octets : le premier contient l'exposant, les suivants la mantisse, poids forts en tête,
— les nombres en double précision (TO 7) sont codés sur 8 octets : le premier contient l'exposant, les suivants la mantisse.
— le descripteur d'une chaîne de caractères comporte 3 octets : le premier indique la longueur de chaîne, les deux autres l'adresse du 1er caractère.
On peut donc atteindre la longueur de la chaîne soit par l'accumulateur B, soit par le descripteur.

A la sortie du module assembleur, il faut également rendre le contenu de l'accumulateur A et du registre X compatible avec le type de la variable qui va recevoir la valeur de sortie.
Dans tous les cas, les registres S et DP doivent être rendus avec les mêmes contenus qu'à l'entrée.

Exemple : Ajouter 256 à un entier.
Après avoir réservé la place (largement suffisante) en ligne 20, on lit les différentes instructions machine, introduites en DATA, et on les dépose en mémoire (lignes 40 à 70).
Pour pouvoir mettre un nombre variable d'instructions en DATA, la dernière valeur, fictive, est supérieure à 255 (&H 100) et sert au test de fin de lecture.
Le module lui-même ne comporte que deux instructions machine (on ne peut faire plus simple) :

```
6C 02       INC 2,X  ; incrémenter en (X)+2
39          RTS      ; retour
```

L'incrémentation de l'octet de poids fort revient à ajouter 256. Notez que pour les entiers, la valeur n'est pas en (X) mais (X)+2.
La nouvelle variable R% reçoit alors la nouvelle valeur qui est affichée avec la précédente.

```
10  ' AJOUTER 256
20  CLEAR ,&H8EFF
30  DEFUSR1=&H8F00
40  ADR=&H8F00
50   READ OCT : IF OCT>255 THEN 80
60   POKE ADR,OCT
70   ADR=ADR+1   : GOTO 50
80  ' UTILISATION
90  N%=10
100 R%=USR1(N%)
110 PRINT N%,R%
120 DATA &H6C,&H02,&H39,&H100
```

Annexe 4 / Liste des codes d'erreurs

Présentation TO 7 - TO 7/7ø

Cette liste alphabétique des codes d'erreurs contient : les deux lettres du code d'erreur, son numéro, le message correspondant aux deux initiales et une explication de cette erreur.

AE 66 Already Exists

Le nom du nouveau fichier dans NAME correspond à un fichier qui existe déjà sur la disquette.

AO 52 Already Open

Le fichier ouvert par OPEN était déjà ouvert.

BD 58 Bad Data

Les données lues dans le fichier par INPUT# ne sont pas celles attendues.

BR 69 Bad Record number

Le numéro d'enregistrement dans GET# ou PUT# vaut zéro ou dépasse le dernier enregistrement du fichier.

BS 9 Bad Subscript

L'indice d'un élément de tableau dépasse la valeur maximum ou bien est négatif.

CN 17 Can't Continue

La commande CONT ne permet pas de poursuivre l'exécution du programme.

DD 1ø Duplicate Definition

Il n'est pas possible de déclarer deux fois le même tableau avec DIM.

DF 65 Directory Full

Il n'y a plus de place sur le catalogue.

DS 56 Direct Statement

Le fichier en cours de chargement contient une commande d'exécution directe.

DU 6Ø Device Unavailable

Le périphérique n'est pas disponible (il n'est pas connecté ou pas allumé).

ER 74 End of Record

La lecture (INPUT#) ou l'écriture (WRITE#) dans un fichier à accès direct, dépasse la fin de l'enregistrement.

FC 5 illegal Function Call

Une fonction a été appelée avec des valeurs interdites.

FD 55 bad File Descriptor

Le descripteur de fichier est incorrect.

FM 51 bad File Mode

Le mode d'utilisation du fichier est incorrect.

FN 23 For without Next

Aucun NEXT n'a été rencontré après l'exécution de FOR.

FO 67 Field Overflow

La somme des longueurs des variables de champ dans un FIELD# dépasse la longueur de l'enregistrement.

FS 7Ø bad File Structure

Les indications concernant les blocs du fichier dans la table d'allocation mémoire ne sont pas cohérentes.

ID 12 Illegal Direct

L'instruction ne peut pas être utilisée en mode direct en dehors d'un programme.

IE 54 Input past End

Il n'est pas possible de lire au-delà de la fin du fichier.

IO 53 Input Output error

Le périphérique ne fonctionne pas bien.

IU 59 device In Use

Le périphérique est déjà en fonctionnement.

LS 15 Long String

Une chaîne de caractères ne peut pas dépasser 255 caractères.

MO 22 Missing Operand

Il manque un opérande dans une opération comme addition, soustraction,...

ND 71 No Disk

Le lecteur de disquettes n'est pas prêt (loquet ouvert).

NE 62 Non Existent file

Le fichier n'existe pas.

NF 1 Next without For

Une instruction NEXT a été rencontrée avant le FOR correspondant.

NO 57 Not Open

Le fichier sur lequel on veut lire ou écrire n'est pas ouvert.

NR 19 No Resume

Il n'y a pas d'instruction RESUME dans la partie qui traite les erreurs (ON ERROR GOTO).

NS 63 No Space

Il n'y a plus aucun bloc libre sur la disquette.

NU 5Ø Not in Use

Le numéro de fichier n'est pas possible.

OB 73 Out of Buffers

Il n'y a plus assez de place dans la zone réservée aux zones tampons des fichiers à accès direct.

OD 4 Out of Data

Il n'y a plus assez de données en DATA pour le READ à exécuter.

OM 7 Out of Memory

Il n'y a plus assez de place en mémoire centrale.

OS 14 Out of String space

Il n'y a plus assez de place en mémoire centrale pour les chaînes de caractères.

OV 6 Overflow

Le nombre à calculer est trop grand.

PP 61 Protected Program

Le programme est protégé et ne peut pas être listé.

RE 2Ø Resume without Error

L'instruction RESUME est exécutée alors qu'il n'y a pas d'erreur.

RG 3 Return without Gosub

L'instruction RETURN a été rencontrée alors qu'il n'y a pas eu de GOSUB.

SE 68 Set to non fielded string **(TO7)**

On ne peut pas faire LSET ou RSET sur une variable qui n'a pas été définie dans un FIELD

SN 2 SyNtax error

Erreur de syntaxe : l'instruction n'est pas connue ou mal rédigée.

ST 16 String formula Too complex

Expression portant sur des chaînes de caractères trop compliquée à traiter.

TF 64 Too many Files

On ne peut pas ouvrir plus de fichiers que le nombre réservé par FILES.

TM 13 Type Mismatch

Il n'est pas possible de mettre un nombre dans une variable chaîne de caractères ou inversement.

UD 76 Unreadable Disk

La disquette n'a pas été initialisée correctement, elle est illisible.

UE 21 Unprintable Error

Erreur sans définition.

UF 18 Undefined user Fonction

La fonction utilisateur USR n'a pas été définie.

UL 8 Undefined Line number

On ne peut pas faire de GOTO ou de GOSUB à une ligne qui n'existe pas.

VF 75 Verification Failure

La relecture après écriture d'un fichier indique une différence.

WP 72 Write Protected

La disquette est protégée en écriture.

/Ø 11 Division par zéro

Une erreur d'exécution de code 7 provoque l'affichage de :
Error 7

Numéro
de l'erreur

1 NEXT sans FOR correspondant
2 Erreur de syntaxe
3 RETURN sans GOSUB correspondant
4 READ sans DATA correspondant
5 Argument incorrect dans une fonction
6 Nombre trop grand dans un calcul
7 Dépassement de capacité en mémoire centrale
8 Numéro de ligne inexistant
9 Indice en dehors des limites
10 Essai de re-dimensionnement d'un tableau
11 Division par zéro
12 Instruction interdite en commande directe
13 Expression ne correspondant pas au type attendu
14 Manque de place pour les chaînes
15 Chaîne trop longue
16 Formule trop complexe
17 Suite de l'exécution impossible
18 Utilisation d'une fonction non définie
19 Instruction RESUME absente
20 RESUME sans ERROR correspondant
21 Erreur non définie
22 Opérande manquant
23 FOR sans NEXT correspondant
50 Numéro de fichier incorrect
51 Mode d'accès au fichier incorrect
52 Fichier déjà ouvert
53 Erreur sur une entrée/sortie
54 Essai de lecture d'un fichier après fermeture
55 Descripteur de fichier incorrect
56 Commande directe dans un fichier en cours de chargement
57 Fichier non ouvert
58 Mauvaises données dans un fichier
59 Périphérique occupé
60 Périphérique absent ou indisponible
61 Programme protégé

62 Le fichier n'existe pas
63 Plus de place sur la disquette
64 Trop de fichiers ouverts en même temps (réservation par FILES)
65 Plus de place dans le catalogue
66 Le nom du nouveau fichier dans NAME correspond à un fichier qui existe déjà sur la disquette
67 FIELD plus grand que la longueur de l'enregistrement.
68 Erreur inexistante sur MO 5
69 Numéro d'enregistrement du PUT ou GET, nul ou supérieur au dernier enregistrement du fichier
70 Impossibilité de retrouver la structure du fichier à partir de la table d'allocation du disque
71 Lecteur non prêt : loquet ouvert
72 Disquette protégée en écriture
73 Manque de place pour les tampons des fichiers à accès direct
74 Lecture au-delà de la fin pour les fichiers à accès direct
75 Erreur d'écriture détectée lors de la vérification
76 Disquette non formattée.

BASIC D.O.S. d'un seul coup d'œil

AUTO 57,105

BACKUP 20,30,105

CIRCLE 33,105
CLOSE 64,106
COPY 28,30,106
CVD 100,106
CVI 100,107
CVS 98,107

DEF FN 55,107
DEF USR 107,129
DEVICE 31,108
DIR 25,108
DRAW 41,109
DSKI$ 110,128
DSKINI 17,111
DSKF 27,111
DSKO$ 111,128

EOF 69,111

FIELD 94,112
FILES 112

GET 53,113
GET # 89,113

HEX$ 55,113

INPUT #65,89,114
INPUT$ 114

KILL 27,114

LINE INPUT # 114
LIST 114
LOAD 22,115
LOAD M 115
LOC 115
LOF 90,115

LSET 94,115

MERGE 24,116
MID$ 56,116
MKD$ 100,116
MKI$ 100,117
MKS$ 98,117
MOTORON 117
MOTOROFF 117

NAME 26,117

OPEN « I » 65,117
OPEN « O » 64,118
OPEN « D » 88,118
OPEN « R » 88,118

PAINT 38,118
PRINT # 80,119
PRINT
#USING 83,119
PUT 53,119
PUT # 88,119

RENUM 57,120
RSET 94,120
RUN 22,121

SAVE 22,121
SAVEM 121
SEARCH 59,122
SKIPE 122
SPACE$ 55,122
STRING$ 55,122
SWAP 57,123

UNLOAD 74,123

VERIFY ON 29,123
VERIFY OFF 29,123
WRITE # 64,88,123

Nota : Les instructions ou fonctions indiquées en italique ne sont pas spécifiques du BASIC disque. Elles ne sont rappelées qu'à propos des fichiers. Pour plus de détails, consultez *Initiation au BASIC TO7* et *Manuel de référence BASIC TO7*.